Vysoká škola báňská – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Návrh a realizace algoritmů nelineárního prediktivního řízení fyzikálního modelu tří nádrží

Design and Implementation of Nonlinear Model Predictive Algorithms for the Three-Tank Physical Model

2020 **Bc. Ondřej Michálek**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Ondřej Michálek**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2612T041 Řídicí a informační systémy

Téma: Návrh a realizace algoritmů nelineárního prediktivního řízení fyzikálního modelu tří nádrží
Design and Implementation of Nonlinear Model Predictive Algorithms for the Three-Tank Physical Model

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Seznámení se s problematikou prediktivního řízení a jeho implementace.
2. Konstrukce fyzikálního modelu tří nádrží včetně aktuátoru, senzoriky a elektronických obvodů.
3. Návrh nelineárního prediktivního řízení na základě matematického modelu soustavy tří nádrží.
4. Ověření navržených řídicích algoritmů v simulaci.
5. Implementace a verifikace navržených řídicích algoritmů pro vytvořený fyzikální model s použitím řídicího systému REXYGEN.
6. Zhodnocení, závěr.

Seznam doporučené odborné literatury:

[1] GRÜNE, Lars a Jürgen PANNEK. *Nonlinear model predictive control.* 2nd Edition. Springer-Verlag, London, 2017, XIV. 456 p. 80 illus., ISBN 978-3-319-46023-9 (hardcover), 978-3-319-46024-6 (eBook). DOI: 10.1007/978-3-319-46024-6.
[2] MACIEJOWSKI, Jan. *Predictive Control with Constraints.* 1 edition. Prentice Hall, 2000. ISBN 978-0201398236.
[3] WANG, Liuping. *Model Predictive Control System Design and Implementation Using MATLAB.* Springer, 2009. ISBN 978-1-84882-331-0.
[4] TAKÁCS, Gergely a Martin GULAN. *Základy prediktívneho riadenia.* Spektrum STU:Bratislava, 2018. ISBN 978-80-227-4826-1.
[5] KRUPA, F., J. NEMCIK, S. OZANA a Z. SLANINA. Educational case study on nonlinear model predictive control. In: *Preprints of the 16th IFAC Conference on Programmable Devices and Embedded Systems,* High Tatras, Slovakia, October 29-31, 2019, s. 459-464.
[6] Firemní dokumentace Matlab and Simulink (MathWorks). [http://mathworks.com]
[7] Firemní dokumentace REXYGEN (Rex Controls, s.r.o.). [http://rexygen.com]
[8] Dokumentace IPOPT. [https://coin-or.github.io/Ipopt/]

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Štěpán Ožana, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 14. *května 2020*

.....
podpis studenta

Poděkování

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Štěpánu Ožanovi, Ph.D. za odbornou pomoc, vedení, konzultace a připomínky, díky kterým se podařilo tuto práci dotáhnout do konce a které mi velmi pomohly.

Abstrakt

Diplomová práce se zabývá problematikou nelineárního prediktivního řízení. Cílem práce bylo navržení a realizace algoritmů nelineárního prediktivního řízení pro fyzikální model tří nádrží, na kterém bude správnost algoritmů ověřena. V úvodní části jsou popsány použité metody a popsána řídicí platforma, která byla použita pro realizaci fyzikálního modelu. Návrh algoritmu nelineárního prediktivního řízení byl proveden na základě matematického modelu soustavy tří nádrží, kde se v prvé řadě navrhla simulace a poté se provedla realizace pro fyzikální soustavu. Simulace byla vytvořena v prostředí Matlab/Simulink, které umožňuje ověřit správnost řešení před samotnou realizací na fyzikálním modelu. Pro realizaci algoritmu regulátoru pro fyzikální model bylo použito také prostředí MATLAB, komunikující přes komunikační rozhraní Rest API s platformou Rexduino, řízenou řídicím systémem REXYGEN. Pro simulaci a realizovaný fyzikální model tří nádrží byla vytvořena vizualizace v integrovaném prostředí řídicího systému REXYGEN.

Klíčová slova

model prediktivního řízení, nelineární model prediktivního řízení, simulace, REXYGEN, MIL, SIL, PIL, Matlab, Rexduino

Abstract

This diploma thesis deals with nonlinear predictive control. The goal of the work was to design and implement algorithms of nonlinear predictive control for the three-tank physics model, where the algorithms will be verified. In the first part of thesis are described used methods and described control platform, which was used for realization of physics model. The design of the nonlinear predictive control algorithm was performed on the basis of a mathematical model of a system of three tanks, where the simulation was first designed and then the realization was performed for the physics model. The simulation was created in Matlab/Simulink, which allows to verify the correctness of the solution before the implementation into the physics model. For realization algorithm for the physics model was used the Matlab environment, which is communicating with Rexduino platform, controlled by REXYGEN controll system, via Rest API communication interface. For simulation and realized physics model of three tanks was created visualization in integrated environment of control system REXYGEN

Key words

Model predictive control, nonlinear model predictive control, simulation, REXYGEN, MIL, SIL, PIL, Matlab, Rexduino

Obsah

Seznam použitých zkratek

APC	Advanced Process Control
DMC	Dynamic Matrix Control
FBD	Function block diagram
GPC	Generalized Predictive Control
GPIO	Processor in the loop
IDCOM	Identification and Command
IDCOM-M	Identification and Command - Mulitvariable
LQG	Linear-Quadratic-Gaussian
LQR	Linear-Quadratic Regulator
MAC	Model Algorithmic Control
MHC	Moving Horizon Control
MIL	Model in the loop
MPC	Model Predictive Control
MPHC	Model Predictive Heuristic Control
NMPC	Nonlinear Model Predictive Control
PIL	Processor in the loop
PWM	Pulse width modulation
QDMC	Quadratic Dynamic Matrix Control
RHC	Receding horizon control
RMPCT	Robust Model Predictive Control
SIL	Software in the loop
SMCA	Setpoint Multivariable Control Architecture
SMOC	Shell Multivariable Optimising Controller

Seznám ilustrací

Úvod

Prediktivní řízení se dnes v průmyslu těší velké popularitě a je jedním z nejvíce rozvíjejících se přístupů v automatickém řízení. Jako jeden z mála teoretických přístupu je i masivně nasazovaný v praxi a v mnoha aplikacích nahrazuje klasické PID regulátory. Důvodem je možnost navržení regulátoru s omezením vstupních, výstupních a stavových veličin. Nejdůležitějším faktorem v prediktivním řízením zůstává princip predikce budoucího chování systému na základě znalosti matematického popisu regulované soustavy. Tím lze dosáhnout kvalitnějšího regulačního pochodu, než je tomu u klasických PID regulátorů. MPC algoritmy jsou aplikovatelné na mnoharozměrné systémy, nestabilní nebo s dopravním zpožděním, což rozšiřuje mnohostranné použití v praxi. S rostoucím výkonem výpočetní techniky je metoda prediktivního řízení stále více nasazována v procesním průmyslu a své uplatnění si začíná nacházet i v energetice a v řízení budov.

Cílem práce je navržení a realizování vhodného algoritmu nelineárního prediktivního řízení (NMPC) pro model tří nádrží a ověření algoritmu implementací na zkonstruovaném fyzikálním modelu. Návrh algoritmu je proveden na základě matematického modelu soustavy tří nádrží. Tento návrh je nejprve simulován a poté realizován pro cílovou platformu, řídicí fyzikální model. Cílova platforma byla zvolena spojení Raspberry Pi s Arduinem řízené řídicím systémem Rexygen, jinak řečeno Rexduino.

Úvodní část práce je věnována seznámení s problematikou spojenou s prediktivním řízením s popisem nejpoužívanějších modelů pro popis řízené soustavy a algoritmů pro prediktivní řízení. Stručně je představena historie a vývoj prediktivního řízení, seznámení s principy a pojmy spojenými s prediktivním řízením jako je horizont predikce, účelová funkce a omezující podmínky. Je zde vysvětlen pojem optimální řízení procesů a popsány rozdíly mezi lineárním a nelineárním prediktivním řízením. A v neposlední části seznámení s hardwarem, a to se zvolenou cílovou platformou, senzorikou, aktuátory a elektronickými obvody.

V následující praktické části je popsán matematický model soustavy tří nádrží, na jehož základě je vytvořený návrh nelineárního modelu prediktivního řízení. Před tím, než bude návrh implementován na fyzikálním modelu, je návrh ověřen simulačními technikami typu MIL, SIL a PIL, aby se ověřila správnost navržených řídicích algoritmů. Simulace MIL je zrealizovaná v simulačním prostředí Matlab na jednom počítači. Simulace SIL je také zrealizována na jednom počítači, ale již s použitím řídicího systému Rexygen, společně se simulačním prostředím Matlab. U simulace PIL je charakteristické simulování matematického modelu v reálném čase a použití řídicího systému na cílové hardwarové platformě, v našem případě Rexduino. Na závěr praktické části se ověří praktické využití algoritmů implementováním na laboratorní fyzikální model. Z důvodu vhodnější prezentace dosažených výsledků je vytvořena vizualizace pro využití modelu jako virtuální laboratoře a jeho využití pro výuku.

Závěr práce je věnován shrnutí dosažených výsledků, porovnání průběhu ze simulace s daty získanými reálným měřením a vytyčení problémů, které vznikly během realizace diplomové práce.

1 Historie a vývoj prediktivního řízení

Myšlenka moderního konceptu teorie řízení se objevila počátkem 60. let 20. století v práci Rudolfa Emila Kálmána v podobě lineárního kvadratického regulátoru LQR. Hlavním úkolem tedy bylo minimalizovat kvadratickou účelovou funkci stavů a vstupů a získat optimální systém s přiměřeně dobrou odezvou pro lineární systémy. Bohužel nebyla v jeho formulaci zahrnuta žádná omezení a nelinearity skutečného systému. Jako regulátor splňuje implicitně řadu požadovaných vlastností systému jako je stabilita, požadovaná šířka pásma a amplitudová a fázová bezpečnost.

Lineárně kvadratické Gaussovské řízení LQG postoupilo ještě o kus dál a bylo definováno jako optimální řízení lineárního systému s ohledem na kvadratickou účelovou funkci s neúplnými měřenými signály poznamenanými bílým šumem. Tento případ se v praxi využíval častěji, jelikož oproti LQR, který byl navržen se zpětnou vazbou od vnitřních stavů, obsahoval LQG zpětnou vazbu od zašuměného výstupu. Ačkoliv se teorie LQG brzy stala standardním přístupem v teorii řízení a existovalo mnoho reálných aplikací, v průmyslovém řízení se tento regulátor nikdy výrazně neuplatnil, a to převážně z důvodu špatných výsledků u nelineárních systémů, nerespektování omezení, nízké robustnosti a složité implementace speciálních požadavků systému.

Samotný vznik prediktivního řízení se v praxi objevil daleko před první publikací na toto téma. Od svého vzniku v 70. letech 20. století se model prediktivního řízení MPC vyvinul z heuristického algoritmu řízení používaného v průmyslu k nové subdisciplíně s teoretickým a praktickým obsahem. V té době z důvodů relativně velkých výpočetních nároků bylo nasazení prediktivního řízení omezeno hlavně na řízení pomalých procesů, například v chemickém průmyslu a rafineriích ropy. Vývoj prediktivního řízení se během několika desetiletí zdokonaloval v oblasti algoritmů pro dosažení lepšího výkonu a stability. Obecně lze tedy vývoj rozdělit na následující generace:

1. Generace MPC (1970-1980)

První popis prediktivního řízení byl představen J. Richaletem a jeho kolegy v roce 1976 a o dva roky později byly poznatky shrnuty a publikovány. Tento přístup byl popsán jako model prediktivního heuristického řízení MPHC (Model Predictive Heuristic Control), který dnes známe jako MAC (Model Algorithmic Control). Softwarové řešení bylo označeno jako IDCOM (IDentification and COMmand). Využívá impulsní odezvu systému k získání matematického modelu a optimalizované vstupy jsou počítány pomocí heurestického iteračního algoritmu. V roce 1980 druhá průkopnická skupina inženýrů ze společnosti Shell Oil, konkrétně Cutler a Remarker, vyvinula vlastní nezávislou technologii MPC v průmyslovém výzkumu. Jednalo se o algoritmus, který bylo možné použít na mnohorozměrné systémy. Řízení se jmenuje DMC (Dynamic Matrix Control) a jako optimalizační metoda je použita metoda nejmenších čtverců. Pro získání popisu soustavy je možné použít impulsní odezvu nebo přechodovou charakteristiku. Počáteční algoritmy IDCOM a DMC představují první generaci technologie MPC. Měly obrovský dopad na řízení průmyslových procesů a sloužily k definování nového smýšlení o průmyslových regulátorech, přestože u těchto prvotních prediktivních regulátorů není zaručena stabilita.

2. Generace MPC (1980-1985)

V roce 1983 popsal C. R. Cutler algoritmus QDMC (Quadratic Dynamic Matrix Control). Oficiální publikace ovšem vyšla až o 3 roky později. Tento algoritmus považovaný za novou, druhou generaci MPC, řeší optimalizaci vstupu numericky pomocí kvadratického programování s omezujícími podmínkami ve tvaru lineárních nerovností. Stále ale není použitelný pro nestabilní systémy, jelikož používá přechodové a impulsní posloupnosti. Algoritmus pracuje pouze s tvrdým omezením, protože formulace měkkých omezení není zcela uspokojivá. Algoritmus QDMC poskytoval systematický přístup k začlenění přísných omezení vstupu a výstupu, ale neexistoval jasný způsob, jak zacházet s neuskutečnitelným řešením. Vývoje tedy postupoval dále.

3. Generace MPC (1985-1990)

V roce 1988 byl publikován společností Setpoint algoritmus IDCOM-M, pro vícerozměrné systémy, proto v názvu připojené M (Mulitvariable). Pro jednorozměrné systémy byla varianta upravena na IDCOM-S. Důležitým faktorem algoritmu IDCOM-M je, že používá dvě samostatné objektivní funkce. Jednu objektivní funkci pro výstupy a pokud existují další stupně volnosti, tak druhou pro vstupy. Téměř s identickou variantou přišla společnost Adersa, a to s algoritmem HEICON. Avšak inženýři ze společnosti Setpoint dále pokračovali ve výzkumu a přišli s algoritmem zvaným SMCA (Setpoint Multivariable Control Architecture). Na konci 80. let inženýři z Francie ze Shell Research přišli s variantou SMOC (Multivariable Optimizing Controller). Právě SMOC pomocí Kalmanova filtru dokáže odhadnout neměřitelné poruchy.

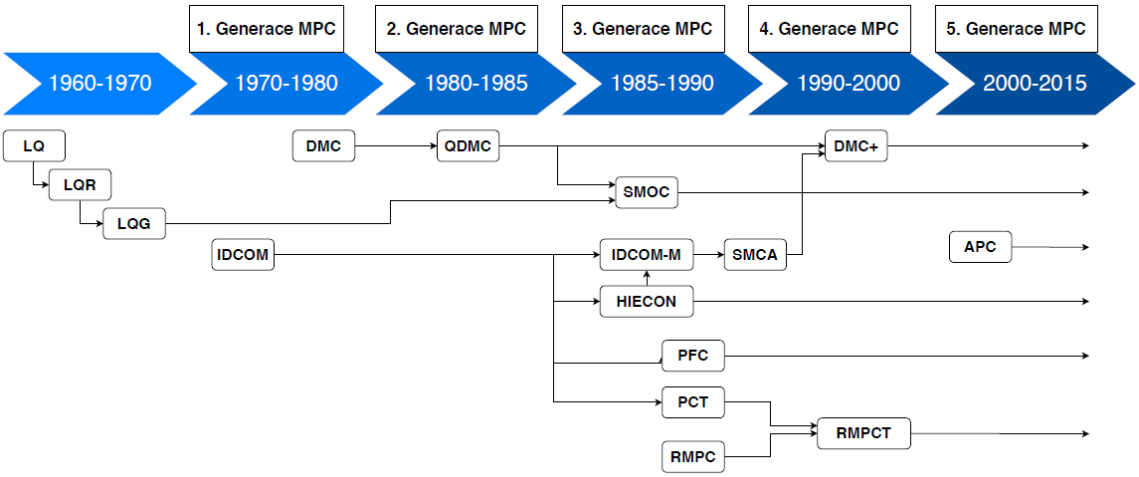
4. Generace MPC (1990-2000)

Čtvrtá generace regulátorů je zaměřená spíše na využití grafického uživatelského rozhraní, explicitní kontrolní objektivní hierarchii a odhad nejistot modelu. Jedná se o produkty uvedené po roce 1995 (2. polovina 90. let) - RMPCT (Robust Model Predictive Control) nebo vylepšené DMC-plus. Také se zvyšuje pozornost teoretiků i praktiků do oblasti nelineárního modelu prediktivního řízení NMPC (Nonlinear Model Predictive Control).

5. Generace MPC (2000-2015)

Poslední pátá generace, se již nezaměřuje na zdokonalení algoritmů, ale na zdokonalení postupů. Cílem je, aby tyto postupy byly plynulejší, rychlejší a snazší pro aplikaci, jak pro vývojáře, tak pro zákazníka. V průběhu desetiletí bylo získáno velké množství znalostí pro online optimalizaci, takže i systémy používané pro dnešní vývoj jsou chytřejší. Pod pátou generaci spadá APC (Advanced Process Control). Tento typ pokročilého řízení procesů zahrnuje několik osvědčených pokročilých řídicích technik.

Níže na obrázku je zobrazen evoluční strom, ilustrující propojení pro nejvýznamnější průmyslové algoritmy v prediktivním řízení procházející skrze jednotlivé generace od počátečního vývoje po současnost. Šipky směřující doprava od počátku do současnosti znázorňují další vývoj tohoto směru.[1][2]



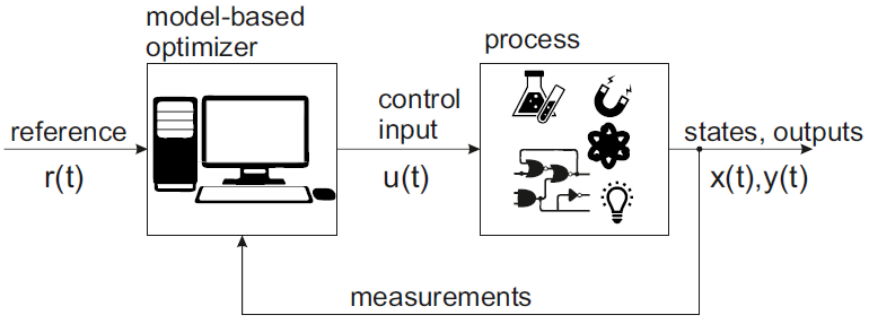
Obrázek 1: *Znázorněný evoluční strom [1]*

2 Úvod do prediktivního řízení

Pojem model prediktivního řízení MPC (Model predictive control), známí také jako MHC (Moving Horiton Control) nebo RHC (Receding Horizon Control) neoznačuje konkrétní algoritmus ani konkrétní strategii řízení, nýbrž škálu metod řízení, které využívají model procesu k získání řídicího signálu minimalizováním objektivní funkce. Různé algoritmy MPC se mezi sebou liší v použití modelu pro reprezentaci procesu a minimalizaci šumů a ztrátové funkce. Všechny tyto konstrukční metody však vedou k návrhu regulátorů, které mají prakticky stejnou strukturu (Obrázek 2.1) a společné znaky jako jsou:

- Používají matematický model soustavy pro predikování budoucích akčních zásahů neboli predikování řízených výstupů systému.
- Známe předem průběh trajektorie referenční veličiny regulované soustavy.
- Ve výpočtu posloupnosti budoucích akčních zásahů je zahrnuta i minimalizace vhodné účelové funkce s budoucími hodnotami přírůstků akčního zásahu a regulační odchylky.
- Ze získané posloupnosti akčních zásahů se provede pouze první akční zásah a zbytek posloupnosti je ignorován. Poté se pro další krok celý postup opakuje s další periodou vzorkování.

V minulosti z důvodu náročnosti na výpočetní výkon, bylo použití prediktivního řízení omezeno pouze na odvětví řízení pomalých procesů, jako například chemický, petrochemický či papírenský průmysl. V dnešní době s vývojem výpočetního výkonu počítačů a řídicích systémů si nalezl široké uplatněné i mezi ostatními průmyslovými procesy vedle klasických PID regulací. Ve většině případů však mají proti klasickým PID regulátorům vyšší kvalitu řízení, dokáží pracovat s mnoharozměrnými systémy a při návrhu lze zahrnout omezení vstupních a výstupních veličin. Jeho obecný princip je stejně dobře aplikovatelná pro lineární i nelineární procesy.[5][10]

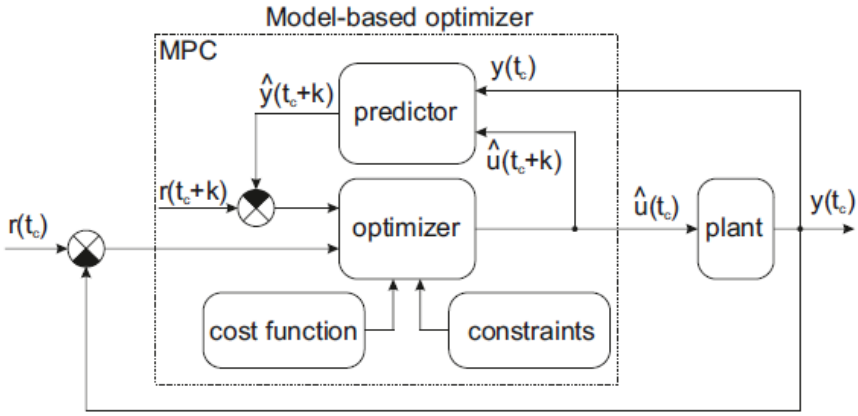


Obrázek 2: *Blokové schéma řízeného procesu*

2.1 Princip prediktivního řízení

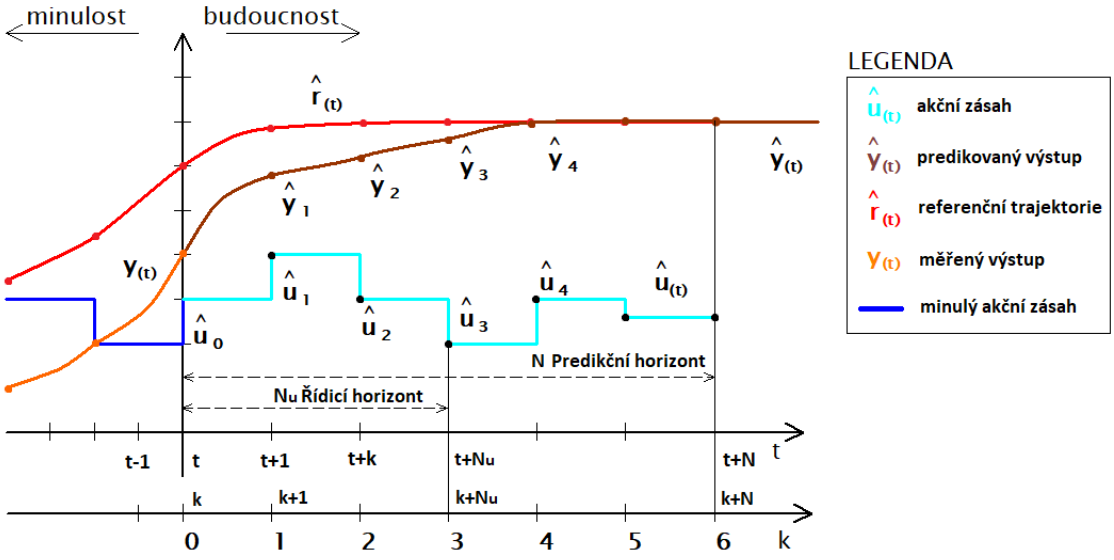
Pro lepší představu obecného principu prediktivního řízení je použitá pro vysvětlení základní struktura MPC, která je na obrázku dole. Matematický model je součástí prediktoru, kde na základě budoucích řídicích signálů a měřených výstupů z reálné soustavy, predikujeme budoucí trajektorii akčních zásahů. Pomocí optimalizátoru, který bere v úvahu účelovou funkci a omezení, se vyřeší

optimalizační problém. Po kterém se aplikuje pouze první akční zásah ze získané posloupnosti akčních zásahů.



Obrázek 3: *Strukturální schéma MPC*

Jak je naznačeno nedílnou součástí prediktivního řízení je dostatečně přesný matematický model reálné soustavy, nikoliv však dokonalý. Právě predikce je závislá na modelu procesu, proto je důležité, aby model správně popisoval dynamické vlastnosti a přesně předpovídal budoucí výstupy procesu pro úspěšný návrh regulátoru. Konkrétní kroky popisující postup prediktivního řízení jsou dobře viditelné na znázorněném obrázku dole. Obrázek reprezentuje MPC s jedním vstupem a výstupem.



Obrázek 4: *Princip prediktivního řízení pro RHC [11]*

- Budoucí výstup $\hat{y}$ pro stanovený horizont N, nazývaný predikční horizont, je predikovány v každém okamžiku t pomocí vhodně zvoleného modelu soustavy. Tyto predikované výstupy $y(t+k)$, kde $k = 1,2,\dots N$ závisí na všech dostupných informací získaných do okamžiku t (minulé vstupy a výstupy) a budoucích řídicích signálech $u(t+k)$, kde $k = 0,1,\dots N-1$. Což jsou řídicí signály, které působí na soustavu.

- Posloupnost neboli trajektorie budoucích akčních zásahů je určena optimalizací stanoveného kritéria, aby se proces udržel co nejblíže referenční trajektorii $r(t+k)$. Toto kritérium je určené obvykle pomocí kvadratické funkce chyby mezi predikovaným výstupním signálem a predikovanou referenční trajektorií. V účelové funkci jsou zahrnuty budoucí predikce výstupů, budoucí trajektorie žádané veličiny a budoucí akční zásahy.
- Přesto že je stanovena posloupnost akčních zásahů, je aplikován pouze první člen akčního zásahu $u(k)$, zatímco další vypočtené řídicí signály jsou ignorovány, protože v dalším vzorkovacím okamžiku je již známo $y(t+1)$ a krok 1 je opakován s touto novou hodnotou a všechny sekvence jsou aktualizovány. Vypočítá se tedy nové $u(t+1)$ pro $u(t+1)$, což se v zásadě bude lišit od $u(t+1)$ v okamžiku t z důvodu nově získaných informací. Tento postup se opakuje, a proto je nazýván strategií pohyblivého horizontu.

2.2 Model řízeného procesu

Jak již bylo zmíněno výše jednou ze základních složek prediktivní model patří mezi základní kameny prediktivního řízení. Pro pochopení a studování systému, je nutné vytvořit model systému. Modely lze rozdělit do dvou kategorií. Jedním je fyzikální model (např. Experimentální jednotka). Tento model vychází z fyzikální nebo geometrické podobnosti mezi modelovaným systémem a modelem, tento model je hmatatelný (fyzický), proto není ideálním použitím pro prediktivní řízení. Druhým je matematický model, tj. Použití matematických rovnic k popisu systému. Tento model je abstraktní, neumožňuje provádět experimenty stejné fyzikální podstaty, ale umožňuje zkoumat jevy na originále pomocí matematického popisu jejich průběhu. Ke studiu systémů je stále více používaný matematický model. Reálné systémy mohou být rozmanité a složité v různých aspektech. Cíle pro analýzu systémů mohou být navíc různé, což odlišuje styly matematického modelu. Metody konstruování matematického modelu lze rozdělit do dvou kategorií.

- Použití fyzikálních principů k vytvoření modelu (principový model). Například pro výrobní proces lze matematický model sestavit podle materiálové bilance, energetické bilance a dalších vztahů. To nejen poskytuje vztah mezi vstupem a výstupem systému, ale také dává vztah mezi stavem a vstupem nebo výstupem systému. Podle tohoto druhu modelu má člověk jasné porozumění systému. Proto je tento druh modelu označován jako „model bílé skříňky“.
- Předpokládejme, že systém vyhovuje určitému druhu matematické rovnice. Změří se vstup a výstup systému. Určí se parametry v modelu pomocí určité matematické metody. Strukturu modelu lze také modifikovat tak, aby se získal vztah mezi vstupem a výstupem systému (identifikační model). Dynamika stavu (tj. Vnitřní dynamika systému) však není známa. Proto je tento druh modelu označován jako „blackbox model“.

Výše uvedené dva způsoby konstrukce modelu lze považovat za dvě třídy předmětů. První metoda může získat podrobný popis systému, ale musí se podrobit důkladnému zkoumání systému, které je předmětem „dynamiky procesu“. Druhá metoda se vyvinula k předmětu „identifikace systému“. Myšlenka MPC není omezena na konkrétní popis systému, ale výpočet a implementace závisí na reprezentaci modelu. V prediktivním řízení lze použít v podstatě libovolný model z těchto dostupných:[4][5]

- stavový popis

- přenosová funkce
- impulzní odezva
- přechodová funkce
- modely pro poruchy
- ostatní modely (spojité modely, neuronové sítě, model popsaný fuzzy logikou)

2.2.1 Stavový popis

Ve vědecké literatuře je model často formulován ve stavovém popisu, protože odvození regulátoru je velmi jednoduché i při popisu vícerozměrných systémů. Formální zápis jednorozměrných a vícerozměrných systémů je totiž identický. Pomocí stavového popisu je snadnější vyjádření stability a kritéria robustnosti. Nevýhodou je znalost stavu systému (matice **A**, **B**, **C**, **D**), pro praxi není tedy příliš vhodný. Je třeba použít stavového pozorovatele. Model je daný popisem:

$$x(k+1) = \boldsymbol{A}x(k) + \boldsymbol{B}u(k) \tag{1}$$

$$y(k) = \boldsymbol{C}x(k) + \boldsymbol{D}u(k) \tag{2}$$

A, B, C, D – matice stavového popisu

$x(k+1)$ – vektor stavu procesu v čase (k)

2.2.2 Přenosová funkce

U tohoto modelu je zapotřebí bezprostřední znalosti řádu polynomu A i B. Nevýhodou použití tohoto modelu je složitější výpočet regulátoru. Využitelnost ale najde pro nestabilní procesy, proto je přenosová funkce v průmyslu více uplatněna než stavový popis systému. Především díky parametrům, charakterizujících přenosovou funkci (například doba zpoždění, zesílení, časové konstanty). Tento model je například použitý v metodě GPC (Generalized Predictive Control). Výstup je závislý na vstupu dle vztahu:

$$A(q^{-1})y(k) = B(q^{-1})u(k) \tag{3}$$

$A(q^{-1}) = a_0 + a_1 q^{-1} + \cdots + a_{nb} q^{-nb}$ - polynom jmenovatele přenosové funkce

$B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_{nb} q^{-nb}$ - je polynom čitatele přenosové funkce

2.2.3 Impulsní odezva

Impulsní odezva nebo také konvoluční model je nejvíce používaná ze všech ostatních modelů v praxi. Je založena na impulsní funkci, kdy systém je vybuzen impulsem (vstup s konečnou délkou) a je změřen výstup. Výstup je tedy závislý na vstupu. U této metody je velkou výhodou, že není třeba znát předchozí znalost systému a že tento model lze snadno získat. Na druhou stranu nevýhodou je, že při větším počtu vzorků N musíme znát větší počet potřebných parametr; a metoda je použitelná pouze pro stabilní procesy.

$$y(t) = \sum_{i=1}^{N} h_i u(t-i) = H(q^{-1})\, u(t) \tag{4}$$

$H(q^{-1}) = h_1 q^{-1} + h_2 q^{-2} + \cdots + h_N q^{-N}$ - polynom n-tého stupně

$y(k)$ – vstup systému

$u(k)$ – výstup systému

$h(i)$ – vzorkovaný výstup

q^{-1} – operátor zpětného posuvu

2.2.4 Přechodová funkce

Je používaná algoritmem DMC a jeho variantami. Metoda je velmi podobná impulsní odezvě s tím rozdílem, že se systém vybudí skokem. Vzhledem k faktu, že impulsní a přechodovou charakteristiku lze jednoduše získat, jsou modely a prediktivní metody na nich založené často používané v průmyslové praxi.

$$y(t) = \sum_{i=1}^{N} g_i \Delta u(t-i) = y_0 + G(q^{-1})(1-q^{-1})\, u(t) \tag{5}$$

$G(q^{-1}) = g_1 q^{-1} + g_2 q^{-2} + \cdots + g_N q^{-N}$ - polynom n-tého řádu

$g(i)$ – vzorkované výstupní hodnoty

$y_0 -$ lze považovat za nulovou hodnotu, aniž by to ovlivnilo výstup

2.2.5 Model poruchy

Výběr modelu reprezentující poruchy je stejně důležitý jako výběr modelu procesu. Často používaným modelem je CARIMA. Model je vhodný použít pro dva typy poruch, náhodné změny, které se vyskytují v náhodných okamžicích (například změny v kvalitě materiálu) a „Brownův pohyb“ a používá se přímo v GPC, EPSAC, EHAC UPC a s malými změnami. Jeho vztah popisuje rozdíl mezi naměřeným výstupem a výstupem vypočteným z modelu:[5]

$$n(t) = \frac{C(q^{-1})}{D(q^{-1})} \xi_s(k) \tag{6}$$

$C(q^{-1}) = c_0 + c_1 q^{-1} + \cdots + c_{nb} q^{-nb}$ – polynom čitatele přenosu poruchy

$D(q^{-1}) = d_0 + d_1 q^{-1} + \cdots + d_{nb} q^{-nb}$ – polynom jmenovatele přenosu poruchy

ξ_s – bílý šum nulového průměru

2.3 Účelová funkce

Účelová funkce bude závislá na volbě zvoleného algoritmu MPC. Pro každý algoritmus prediktivního řízení se navrhuje jiná účelová funkce k získání optimálního budoucího akčního zásahu. Nejčastěji se pro výpočet kritéria kvality regulace používá kvadratické kritérium. Pro obecný popis se uvádí zjednodušená účelové funkce ve formátu:

$$J_N(N_1, N_2, N_u) = \sum_{k=N_1}^{N_2} \delta(i)[(\hat{y}(t+i) - w(t+i)]^2 + \sum_{i=1}^{N_u} \lambda(i)[\Delta u(t+i-1)]^2 \quad (7)$$

Formulace obsahuje závislost výstupního signálu na kvadratické regulační odchylce a kvadratické změně přírůstku akční veličiny. Cílem je, aby budoucí výstup v uvažovaném horizontu sledoval referenci a současně aby byla splněna penalizace. Některé metody neberou druhý

N_1 a N_2 – jsou minimální a maximální predikční horizonty

N_u – je řídicí horizont

$\delta(i)$ – penalizační koeficient regulační odchylky

$\lambda(i)$ – penalizační koeficient změny akčního zásahu

$\hat{y}(t+i)$ – predikované výstupy

$w(t+i)$ – budoucí referenční hodnota

$\Delta u(t+i-1)$ – budoucí změny akčního zásahu

Význam parametrů N_1 a N_2 je poněkud intuitivní. Označují meze instancích, ve kterých požadujeme sledovat referenční veličinu. Pokud se tedy nastaví vyšší hodnotu N_1 budou se ignorovat vyskytnuté chyby v prvních okamžicích. Výsledkem bude plynulá reakce procesu a vyhneme se problémům s neminimálně fázovými procesy. N_1 volíme minimálně $T_d + 1$, kde T_d je přibližná hodnota dopravního zpoždění. Pokud není známe dopravní zpoždění volíme obvykle hodnotu 1. Maximální horizont N_2 by se měl zvolit tak, aby dokázal pokrýt co nejvíce důležitou část přechodové charakteristiky. Ideální nastavení bývá zvolení hodnoty, kdy výstupní veličina přejde z 10 % na 90 % své ustálené hodnoty. Řídicím horizontem N_u můžeme ovlivnit výpočetní náročnost systému. Pro jednoduché zařízení bude 1 pro vyšší hodnoty se budou zvyšovat i aktivity výpočtů. Doporučuje se volit poloviční hodnota vzhledem k maximálnímu predikčnímu horizontu. Musí tedy platit podmínka:

$$N_u \leq N_2 \quad (8)$$

Koeficienty $\delta(i)$ a $\lambda(i)$ jsou sekvence, které zohledňují budoucí chování. Volí se jako konstanty nebo ve formě exponenciálních vah. Penalizační koeficient $\lambda(i)$ umožňuje v účelové funkci nastavit, zda bude mít větší váhu regulační odchylka nebo změna přírůstku akční veličiny. Pokud bude:

- $\lambda(i) < 1$ rozkmitá se akční veličina, ale docílíme k lepšímu sledování trajektorie žádané veličiny.
- $\lambda(i) > 1$, pak akční zásah se bude zmenšovat, avšak kvadratická odchylka se zvýší.

Obvykle je ale stejně jako penalizační koeficient $\delta(i)$ nastaven na hodnotu 1. V případě, kdy má však nějaká regulovaná veličina větší prioritu než regulační odchylka, která má dosahovat menších hodnot vzhledem k ostatním regulačním odchylkám, pak můžeme hodnotu koeficientu u této veličiny zvýšit a ostatní snížit k 0.[4][10][13]

2.4 Omezující podmínky

Zahrnutí omezujících podmínek do návrhu prediktivního řízení je jednou z jeho předností, proto se tolik rozšířil v průmyslu. Každý proces obsahuje určitá omezení, ať již se omezení týkají například senzorů, aktuátorů, omezení v technologii či bezpečnosti procesu apod. V tomto případě se bavíme o fyzických omezení, kde například omezujeme vstupní veličiny procesu pracující v nastaveném rozsahu, jako je například průtok kapaliny. Tato omezení označujeme jako tvrdá omezení, která nesmíme překročit.

Nejjednodušší aplikování omezení a v praxi nejpoužívanější způsob realizace je analytické řešení optimalizační úlohy bez uvážení omezení a následné aplikování omezení na jednotlivé veličiny. V takovém případě ale nasazujeme omezení pouze na výstupní veličiny nebo akční zásah, které vystupují z algoritmu optimalizace. Tím se sice může docílit funkčnosti, ale v jiných případech, kdy je potřeba zajistit optimalizaci dle stanovených kritérií, musíme zahrnout omezení přímo do výpočtu optimalizace. V jiném případě by mohlo také dojít k tomu, že regulátor při řízení soustavy nebude mít potřebné informace o omezení, a tudíž bude zvyšovat akční zásah s cílem dosáhnout referenční hodnoty, které nedosáhne v souvislosti nasazeným omezením. Docházelo by tak k nežádoucímu regulačnímu průběhu.

Ideální volbou je zahrnutí omezujících podmínek do výpočtu optimalizační úlohy. Takto lze omezit nejen veličiny vystupující z algoritmu, ale také výstupní veličiny nebo vnitřní stavy, pokud je použit stavový model soustavy. Problém je řešen kvadratickým programováním.[10][12]

Typy omezení dělíme na:

- **Měkká omezení** (soft constraints) – Mohou být za určitých podmínek porušena, ale tato situace je následně penalizována (uměle definovaná omezení, dočasné navýšení zátěže motoru). Jsou zavedena z důvodu nevyhovujících tvrdých omezujících podmínek, kdy zrovna nelze nalézt vhodné optimální řešení z technologického či jiného hlediska.
- **Tvrdá omezení** (Hard constraints) – Nesmí být ze svého vyhrazené prostoru překročena, jedná se o fyzické omezení procesů. Tvrdá omezení se obvykle používají u vstupů systému (omezení změny akčního zásahu, omezení akčního zásahu). Naopak u stavů a výstupů systému je doporučeno používat měkká omezení. Narozdíl od vstupů systému, které definuje regulátor jsou stavy a výstupy systému předmětem měření a jsou tedy zatíženy šumem. Snadno by tak mohlo dojít k překročení pevného omezení, což by vedlo k neřešitelnosti úlohy optimalizace.[26]

Mezi běžná omezení patří například:

Omezení stavové veličiny

$$x_{min} \leq x(k) \leq x_{max} \tag{9}$$

x_{min} – minimální hodnota stavové veličiny

x_{max} – maximální hodnota stavové veličiny

Omezení výstupní veličiny

$$y_{min} \leq y(k) \leq y_{max} \tag{10}$$

y_{min} – minimální hodnota stavové veličiny

y_{max} – maximální hodnota stavové veličiny

Omezení velikosti akčního zásahu

$$u_{min} \leq u(k) \leq u_{max} \tag{11}$$

u_{min} – minimální hodnota stavové veličiny

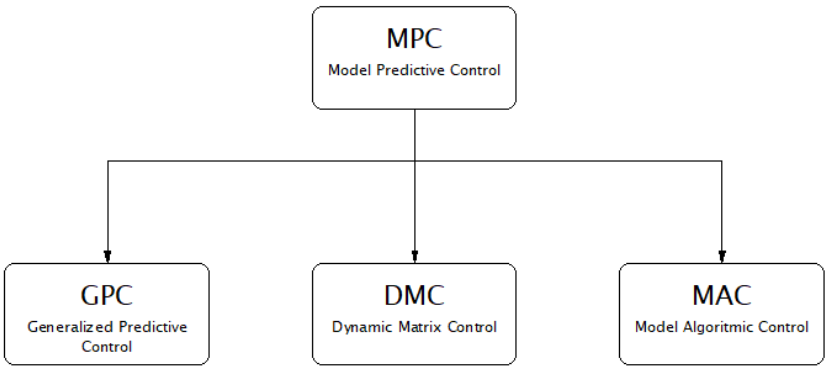
u_{max} – maximální hodnota stavové veličiny

Omezení rychlosti změny akčního zásahu

$$z\Delta u_{min} \leq u(k) - u(k-1) \leq \Delta u_{max} \tag{12}$$

2.5 Algoritmy prediktivního řízení

Mezi nejoblíbenější algoritmy spojeným s modelem prediktivního řízení jsou Dynamic Matrix Control (DMC), Model Algorithmic Control (MAC) a Generalized Predictive Control (GPC). I když se tyto algoritmy v určitých detailech liší hlavní myšlenka je velmi podobná. MPC je ve své základní neomezené formě úzce spjata s lineárním kvadratickým řízením. V případech MPC s omezujícími podmínky vede k řešení optimalizačního problému, který je řešen online v reálném čase v každém vzorkovacím intervalu.



Obrázek 5: *Základní algoritmy pro MPC*

2.5.1 GPC

GPC je nejpopulárnějším algoritmem prediktivního řízení. Je uznáván jak v akademické teorii řízení, tak v průmyslové praxi. Algoritmus byl vyvinut při zkoumání adaptivního řízení. GPC nejen zdědí výhody adaptivního řízení pro jeho použitelnost ve stochastických systémech, on-line identifikaci,

ale zachovává si výhody prediktivního řízení pro jeho optimalizaci ustupujícího horizontu (RHC), nižšího požadavku na přesnost modelování atd. Metoda je aplikovatelná na systémy s proměnným dopravním zpožděním, systémy vyššího řádu a nestabilní systémy v otevřené smyčce. GPC má následující charakteristiky:

- Spoléhá se na tradiční parametrické modely. V modelu systému je tedy méně parametrů. Metody MAC i DMC používají neparametrické modely, tj. Model impulzní odezvy.
- Byla vyvinuta při vyšetřování adaptivního řízení. Zdědí jeho výhody, ale je robustnější.
- Využívá techniky vícestupňové predikce, dynamické optimalizace a korekce zpětné vazby. Proto je řídicí efekt lepší a vhodnější pro průmyslové procesy.[4]

2.5.2 MAC

Základem bylo vyvinout nový řídicí algoritmus, který má nižší požadavky na znalost modelu, zahrnující pohodlný on-line výpočet a má lepší výkon řízení. Ve snaze docílit těchto požadavků pro průmyslové procesy vznikl řídicí algoritmus MAC, Prediktivní řízení s tímto algoritmem proto není teoretickým produktem, nýbrž produkt vyvinutý z praxe. Metoda je založena na modelu impulsní odezvy, algoritmus nepočítá se změnou akčního zásahu, ale s celým akčním zásahem. MAC se také nazývá prediktivním heuristickým řízením (MPHC). Odpovídajícím průmyslovým softwarem je IDCOM (Identification-Command). MPHC měl ve své době tyto jedinečné vlastnosti:

- Mnoharozměrný proces, který má být řízen, je reprezentován modelem impulsivní odezvy, která tvoří interní model (tj. Model uložený v počítačové paměti). Tento model se používá pro predikci on-line a jeho vstupy a výstupy jsou aktualizovány podle skutečného stavu procesu. Ačkoli to bylo možné identifikovat on-line, interní model je většinu času počítán off-line.
- Strategie je stanovena pomocí referenční trajektorie, která definuje chování soustavy v uzavřené smyčce. Tato trajektorie je zahájena na skutečném výstupu procesu a má sklon k požadované žádané hodnotě. Jedním z hlavních ladících knoflíků MPHC je referenční trajektorie.
- Regulátory jsou počítány postupem, který je v obecném případě heuristický. Budoucí vstupy jsou počítány tak, že při použití na rychle interní prediktivní model indukují výstupy co nejblíže k požadované referenční trajektorii.[4]

2.5.3 DMC

DMC má mnoho podobností s MAC. Je to algoritmus založený na modelu přechodové funkce, který je ekvivalentní s modelem impulzní odezvy. Předpokládá se, že proces je stabilní a že predikce poruchy je konstantní po celé délce horizontu. DMC však používá inkrementální algoritmy, což je velmi efektivní při odstraňování chyby v ustáleném stavu. Ve srovnání s DMC má MAC své výhody, jako je vyšší schopnost potlačení rušení. Metoda je obzvláště vhodná pro systémy s více vstupy a výstupy a výhodou je i doplnění omezení procesních veličin při řešení minimalizačního kritéria. Ve skutečných aplikacích závisí výběr DMC a MAC na přesné situaci. Až dosud je DMC v procesním průmyslu nejrozšířenější. Nevýhodou je nevhodnost použití pro řízení nestabilních systémů a velký počet parametrů nutný k popisu procesu.[4]

3 Problém optimálního řízení

Optimalizační úlohy vznikají typicky při řešení praktických úloh, když chceme dosáhnout co nejlepšího výsledku v rámci našich možností. Například v informatice je optimalizace takový proces k nalezení minima kriteriální funkce, která vede k jeho vyšší efektivitě nebo ke snížení nároků celého řídicího systému. Nejlepší výsledek posuzujeme pomocí účelové funkce. Snažíme se najít takové kombinace hodnot optimalizačních proměnných, pro níž daná účelová funkce nabývá svého maxima, respektive minima. Standartní tvar optimalizačního problému lez formulovat jako:

$$maxf(x) \tag{13}$$

$$h(x) = 0 \tag{14}$$

$$g(x) \leq 0 \tag{15}$$

$$x_{min} \leq x \leq x_{max} \tag{16}$$

Kde

$f(x)$ je účelová funkce

$h(x)$ je rovnostní funkce omezení.

$g(x)$ je nerovnostní funkce omezení,

Optimalizační úlohy, řeší ekonomické nebo technické problémy, lze je používat v řadě průmyslových odvětvích jako je optimální plánování výroby, optimální spalovací poměr (vzduch/palivo) nebo optimální využití zásob. V oblasti řídicích a informačních systémů a automatizační technice nachází využití v oblasti jako optimální nastavení konstant regulátorů, optimalizace struktury regulátorů a regulačních obvodů či volba nelineárních optimálních korekčních prvků nebo optimální strategie při rozhodovacích úlohách (rozpoznávání obrazů, předmětů, aj.).

Přehled metod řešení optimalizačních problémů

Existují obecné metody, které lze aplikovat u mnoha optimalizačních úloh i zdánlivě velmi odlišných. Metody řešení optimalizačních problémů lze rozdělit do těchto základních skupin:

- Klasické metody hledání extrémů funkcí
- Lineární programování (LP)
- Nelineární programování (NLP)
- Variační počet
- Pontrjaginův princip maxima (PMP)
- Dynamické programování

Používají se zejména numerické metody řešené pomocí počítače než analytické metody, to je způsobeno použitím různých omezení či doplňujících podmínek. Pokud bychom chtěli tyto metody

rozdělit na základní rozdělení, tak nejúčelnější rozdělení optimalizačních úloh je dle typu matematické formulace optimalizační úlohy:

- Statická optimalizace
- Dynamická optimalizace

Při problému statické optimalizace není nalezené řešení funkcí času. To si lze představit tak, že hledané řešení odpovídá jednomu konkrétnímu časovému okamžiku, případně ustálenému stavu. Oproti tomu u problému dynamické optimalizace představuje nalezené řešení funkce závislé na čase.

V další části se budu věnovat metodám, které se primárně týkají úloh statické optimalizace, a to lineárnímu a nelineárnímu programování. Úlohy dynamické optimalizace jdou však rovněž řešit metodami statické optimalizace, a to zejména metodami nelineárního programování (NLP).

3.1 Lineární programování

Aplikace této metody se poprvé pokusily aplikovat v oblasti výrobních plánů a ekonomie koncem 30. let. Během druhé světové války bylo lineární programování široce využíváno k řešení dopravy, plánování a přidělování zdrojů podléhající určitým omezením jako je například dostupnost. Lineární programování řeší optimalizační úlohy pospané soustavou lineárních rovnic a nerovnic. Účelová funkce a omezující podmínky jsou také v lineárním tvaru. Úlohy lineárního programování lze provést různými metodami:

- simplexovou metodou
- numerickými metodami
- grafickým řešením

Maticový zápis lineárního programování:

Hledá se minimum $\boldsymbol{x}^*$ účelové funkce $J = f(\boldsymbol{x})$ tak, aby byly splněny předepsané podmínky ve tvaru lineárních nerovností (18), lineárních rovností (19) a omezení (20).

Matematická formulce:

$$\boldsymbol{x}^* = \min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{17}$$

Za splněných podmínek:

$$\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b} \tag{18}$$

$$\boldsymbol{A}_{eq} \cdot \boldsymbol{x} = \boldsymbol{b}_{eq} \tag{19}$$

$$\boldsymbol{l}_b \leq \boldsymbol{x} \leq \boldsymbol{u}_{\boldsymbol{b}} \tag{20}$$

3.2 Nelineární programování

Ačkoliv aplikace lineárního programování funguje v mnoha případech dobře, v některých situací problém nelze přesně vyřešit, aniž by byly zahrnut nelineární komponenty. Právě nelineární programování řeší optimalizační úlohy, u kterých je účelová funkce nebo omezující podmínky předepsány nelineárními funkcemi. Omezující podmínky mohou být ve tvaru rovnic či nerovnic.

Maticový zápis lineárního programování:

Hledá se minimum $\boldsymbol{x}^*$ účelové funkce $J = f(\boldsymbol{x})$ tak, aby byly splněny předepsané podmínky ve tvaru lineárních nerovností (22), lineárních rovností (23), omezení (24), nelineárních nerovností (25) a nelineárních rovností (26).

Matematická formulce:

$$\boldsymbol{x}^* = \min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{21}$$

Za splněných podmínek:

$$\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b} \tag{22}$$

$$\boldsymbol{A}_{eq} \cdot \boldsymbol{x} = \boldsymbol{b}_{eq} \tag{23}$$

$$\boldsymbol{l}_b \leq \boldsymbol{x} \leq \boldsymbol{u}_{\boldsymbol{b}} \tag{24}$$

$$c(\boldsymbol{x}) \leq 0 \tag{25}$$

$$c_{eq}(\boldsymbol{x}) = 0 \tag{26}$$

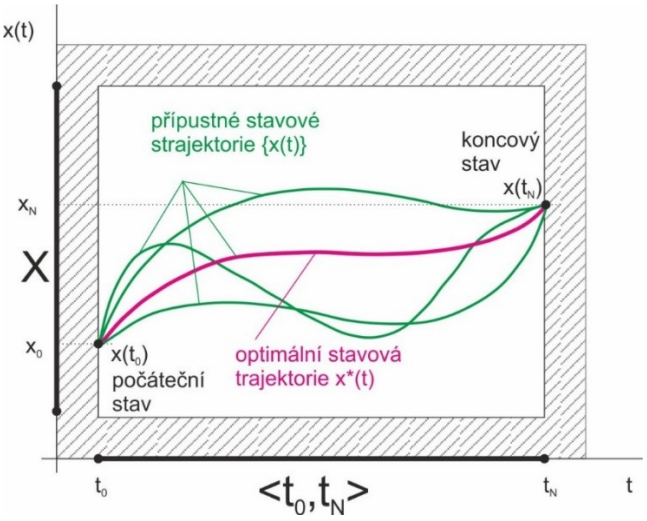
V některých případech mohou některé nebo všechny omezující podmínky chybět, jindy mohou být některé podmínky lineární. Je-li účelová funkce kvadratická, mluvíme o tzv. kvadratickém programování (omezující podmínky se zde obvykle uvažují lineární).

3.3 Úlohy optimálního řízení

Uvažujme dynamický systém:

$$\dot{\boldsymbol{x}} = f[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \tag{27}$$

Pro který hledáme takové řízení $\boldsymbol{u}(t)$, aby systém převedlo z daného počátečního stavu do daného koncového stavu. Řízení, které tento požadavek splní, nazýváme jako přípustné řízení. Přípustných řízení může existovat více, resp. nekonečně mnoho. Stavové trajektorie způsobené tímto řízením nazýváme jako přípustné trajektorie, viz Obrázek dole. Z množiny přípustných trajektorií pak lze vybrat tzv. optimální trajektorii, která splňuje speciální podmínky, viz níže.



***Obrázek 6**: Dynamické optimalizace-přípustné trajektorie a optimální trajektorie [32]*

3.3.1 Formulace úlohy optimálního řízení

Cílem optimálního řízení je najít takové přípustné řízení $\boldsymbol{u} = \boldsymbol{u}^*$, které převede systém z počátečního stavu $\boldsymbol{x}_0$ do koncového stavu $\boldsymbol{x}_N$ tak, aby se minimalizovalo zadané kritérium J, a aby byly splněny všechny omezující podmínky. [32]

Základní úloha optimálního řízení je zadána následovně:

Pro dynamický systém, který má být řízen:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, t) \tag{28}$$

$$\boldsymbol{y} = g(\boldsymbol{x}, \boldsymbol{u}, t) \tag{29}$$

je definován počáteční stav:

$$\boldsymbol{x}_0 = \boldsymbol{x}(t_0) \tag{30}$$

a omezení (podmínky):

$$[\boldsymbol{x}(t), t] \in K \; pro \; t = t_N \tag{31}$$

$$\boldsymbol{x}(t) \in X \; pro \; t \in \langle t_0, t_N \rangle \tag{32}$$

$$\boldsymbol{u}(t) \in U \; pro \; t \in \langle t_0, t_N \rangle \tag{33}$$

Minimalizuje se objektivní funkce vyjádřená funkcionálem integrálního typu:

$$J = \Phi(\boldsymbol{x}_N, t_N) + \int_{t_0}^{t_N} L(\boldsymbol{x}, \boldsymbol{u}, t)\mathrm{d}t \tag{34}$$

Tento tvar je znám jako Bolzův tvar. Je to obecný tvar, ve kterém můžeme identifikovat dva dílčí členy:

Φ - statická část (Mayerův člen), popisuje a penalizuje chování v koncovém stavu

L - dynamická část (Lagrangeův člen), popisuje a penalizuje chování na časovém intervalu

3.3.2 Metody optimalizace

Spojitý problém optimálního řízení (OCP) lze řešit těmito metodami:

- Dynamické programování
- Nepřímé metody
- Přímé metody

3.3.2.1 Dynamické programování

Dynamické programování je velmi obecná optimalizační metoda, která je založena na principu optimality formulovaném Bellmanem, považovaného za zakladatele dynamického programování. Problém, který chceme řešit dynamickým programováním se nejprve rozdělí do jednodušších podúloh celkového problému. Celkové řešení pak získáme složením jednotlivých podúloh.

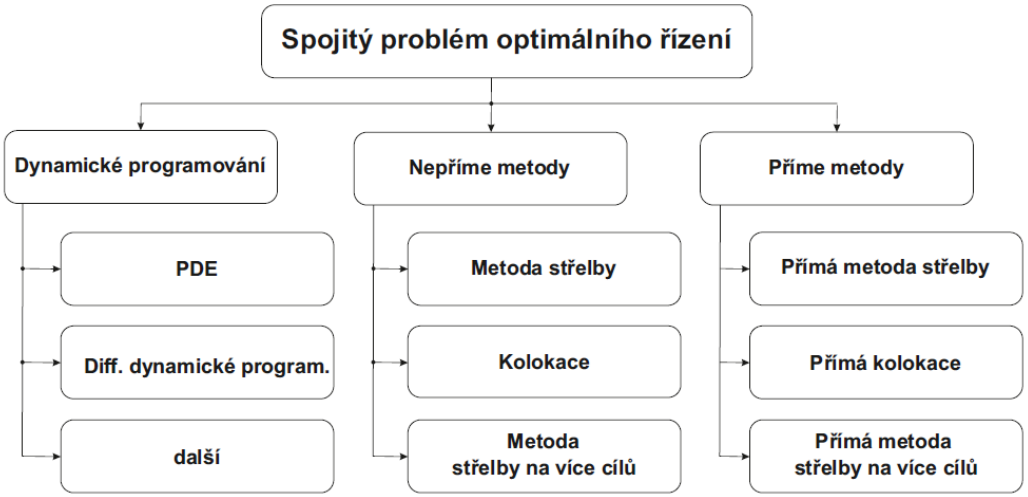
3.3.2.2 Nepřímé metody

Hlavní myšlenkou je převod úlohy OCP na okrajový problém BvP (boundary value problem). Po převedení OCP na BvP získáme více omezujících podmínek, než je počet stavů. Taková úloha se pak stává předurčenou jinak řečeno:

$$(počet\ podmínek\ -\ počet\ stavů\ =\ počet\ volných\ parametrů).$$

3.3.2.3 Přímé metody

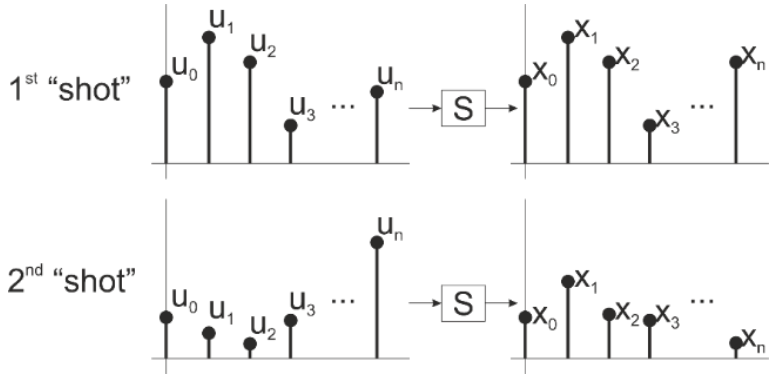
Základní myšlenkou je převod OCP na NLP. Spojitý problém optimálního řízení (OCP) je převeden na problém nelineárního programování (NLP) pomocí diskretizace veličin v čase. Tímto přístupem je možno pro nalezení řešení problému dynamické optimalizace použít metody pro řešení úloh statické optimalizace. V diplomové práci se zaměřuji právě na tuto metodu, konkrétně používám přímou metodu střelby.



Obrázek 7: *Diagram přehledu metod pro řešení OCP*

Přímá metoda střelby

Metoda předpokládá diskretizaci řízení a stavových veličin v čase na intervalu, kde je hledáno řešení. Pro příslušnou kombinaci hodnot akčních zásahů přísluší po průchodu soustavou určitá kombinace hodnot stavových veličin. Při další iteraci "výstřelu" algoritmus přizpůsobí hodnoty akčních zásahů pro splnění účelové funkce. V účelové funkci se vyskytuje řízení, stavy a omezující podmínky. Celý proces se opakuje tak dlouho, dokud není nalezeno řešení s dostatečnou požadovanou přesností.[32]



Obrázek 8: *Přímá metoda střelby*

3.4 Nelineární prediktivní řízení

Základní problém přístupu NMPC je v tom, že implementační platforma musí být schopna vyřešit omezený optimalizační problém ve stanovené lhůtě. Čas pro optimalizaci se snižuje se zvyšováním rychlosti dynamiky řízené soustavy. V důsledku toho byla implementace NMPC doposud obecně omezena na procesy s pomalou nebo jinak velmi jednoduchou dynamikou. Omezení výpočetním výkonem řídicího systému pro výpočet řešení bylo tak překážkou pro využití na rychlejší nebo složitější procesy. S rostoucím výpočetním výkonem počítačů, řídicích systému atd, přehodnocuje využití prediktivního řízení v praxi i výzkumu pro rychlejší procesy.

V této podkapitole však není popsán kompletní přehled existujících technik NMPC. Poskytuje hlavně souhrn obecných informací pro vytvoření představy o daném tématu a ukazuje princip na nichž je NMPC založeno. Dále nastiňuje hlavní výhody či nevýhody NMPC a ukazuje některé teoretické, výpočetní a implementační aspekty. Úloha nelineárního prediktivního řízení řešena pomocí RHC, lze považovat za podkategorii obecné úlohy optimálního řízení, protože u této metody řešíme optimální úlohu na otevřené smyčce na predikčním horizontu. [14][15]

Lineární vs. nelineární prediktivní řízení

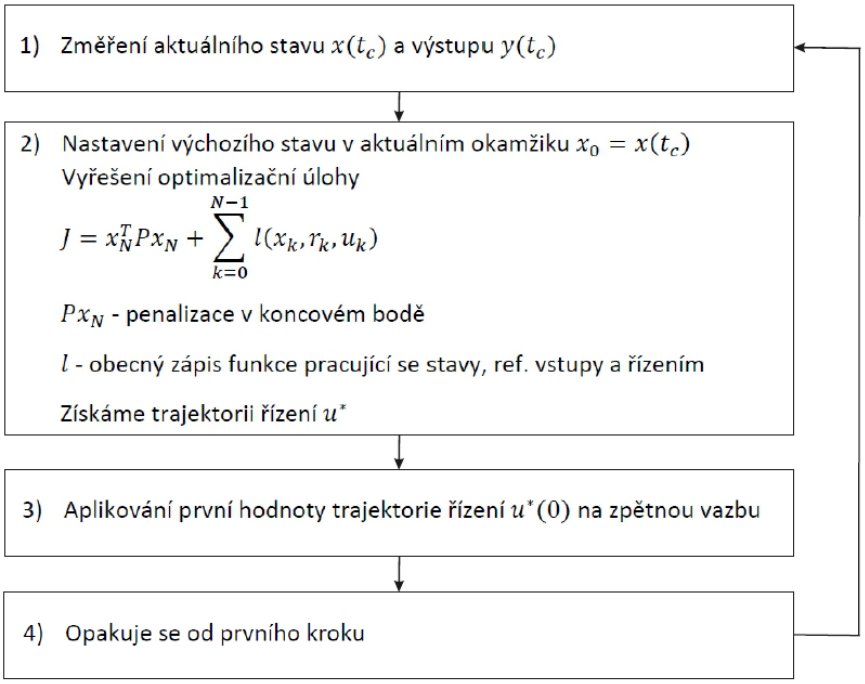
Lineární MPC bylo za poslední desetiletí vyvinuto do fáze, kdy dosáhlo dostatečné zralosti, aby se obrátila pozornost aktivního výzkumu na nelineární řízení. V 90. letech minulého století došlo k zvýšení pozornosti teoretiků i praktiků na oblast prediktivního řízení nelineárních modelů NMPC. Zájem o tuto oblast je dán skutečností, že mnoho systémů je obecně neodmyslitelně nelineárních a dnešní procesy musí být provozovány za přísnějších výkonových specifikací. Současně je s narůstajícím nasazením na složitější procesy třeba splnit stále více omezení, například z hlediska životního prostředí a bezpečnosti. V těchto případech jsou lineární modely často nedostatečné k popisu dynamiky procesu a je třeba použít nelineární modely. Toto motivuje k použití prediktivního řízení nelineárního modelu.

Teoreticky je rozšíření principů na nelineární případy totožné, ale v praxi se objevuje několik problémů. Jedním z nich je obtížné získání výstižného a výpočetně efektivního nelineárního modelu. Nelineární procesy disponují nedostatkem technik pro identifikaci, proto získání modelu z prvních principů není vždy proveditelný. Dále optimalizační problém může být nekonvexní, řešení může být tak obtížnější než u kvadratického programování.

V případě lineárního MPC lze predikovanou trajektorii stavu a výstupu přímo vyjádřit jako lineární funkci současného stavu a vstupní trajektorie. Pokud je účelová funkce definována jako lineární nebo kvadratická ve vnitřních stavech, výstupech či vstupech, výsledným optimalizačním problémem je lineární programování (LP) nebo kvadratické programování (QP). Lineární programování i kvadratické programování lze efektivně řešit pomocí vysoce výkonných solverů (softwarových balíčků pro řešení). Díky tomu je použití lineárních MPC v reálném čase možné použít i u dnešních počítačů a mikrokontrolérů i pro krátkou dobu vzorkování.

U NMPC znemožňuje nelineární dynamika systému vyjádřit přesnou predikci jako lineární funkci aktuálního stavu a vstupní trajektorie. Takže i kdyby byla účelová funkce lineární nebo kvadratická, což bývá běžný případ, tak výsledný optimalizační problém by byl nelineární. Tento problém s nelineární optimalizací řešíme jako problém nelineárního programování (NLP). Nelineární programování je subdisciplína matematického programování, řešící problém nalezení minima nebo maxima nelineární funkce. Je vysoce náročná na výpočetní výkon, proto je třeba použít zcela odlišný přístup pro optimalizaci. Účelová funkce by měla být formulována tak, aby její minimalizace vedla ke splnění požadovaného řízení, jako je sledování reference atd. [15]

Schéma řešení úlohy NMPC je následující:



Obrázek 9: *Schéma řešení NMPC*

4 Fyzikální model tří nádrží

Jedná se o soustavu vzájemně propojených tří válcových skleněných nádrží. Do levé části je seshora přiváděn přítok vody, čímže je nádrž postupně plněna. Zároveň je v dolní části levé nádrže realizován odtok do prostřední nádrže. Z této prostřední nádrže je realizován odtok do vedlejší pravé nádrže. Z pravé nádrže je realizován volný odtok ven ze soustavy do společné nádobky na vodou, ze které jsou skleněné nádoby plněny a tvoří tak uzavřený okruh.

Propojovací i ukončovací trubička má po celém svém profilu vnitřní průměr $6\,mm$ na tloušťce, na vnějším průměru nezáleží. Válcové nádrže mají vnitřní průměr $42{,}5\,mm$ na tloušťce, zde na vnějším průměru opět nezáleží. Výška válcových nádrží je $392\,mm$, při čemž potřebujeme pro návrh k dispozici výšku pouze $300\,mm$, takže máme dostatečnou rezervu. Vzájemná vzdálenost nádrží mezi sebou by měla být co nejmenší.

Obrázek 10: *Reálná soustava tří nádrží*

Parametry modelu:

Vnitřní průměr nádoby: $d = 0{,}0425\,m$

Vnitřní průměr trubičky mezi nádobami: $d_v = 0{,}005\,m$

Plocha hladiny v nádobě: $S = \frac{\pi.d^2}{4} = 0{,}0015\,m^2$

Společní plocha propojující nádoby: $S_v = \frac{\pi.{d_v}^2}{4} = 0{,}00002874\,m^2$

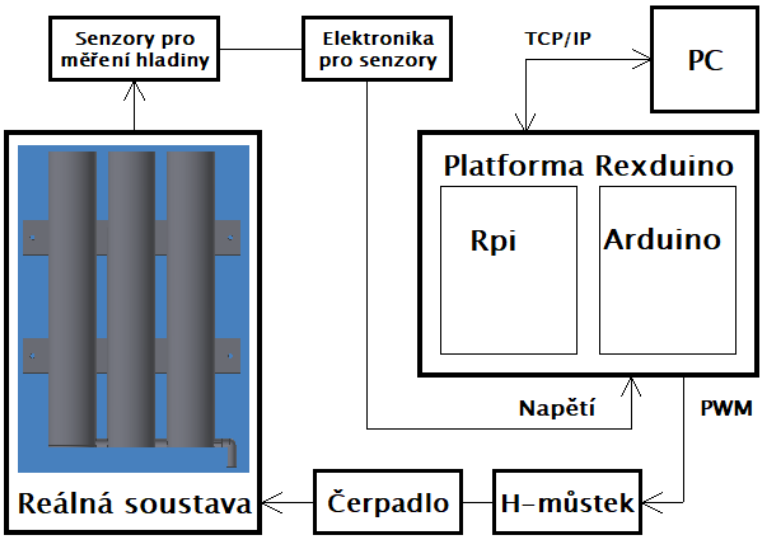
Průměr potrubí výtok: $d_{v3} = d_v = \ 0{,}005\,m$

Plocha výtoku ze třetí nádoby: $S_v = \frac{\pi.{d_{v3}}^2}{4} = 0{,}00002874\,m^2$

Gravitační zrychlení: $g = 9{,}81\ \frac{m}{s^2}$

4.1 Realizace reálné soustavy

Jedním z úkolů bylo navržení a realizace fyzikálního modelu tří nádrží, včetně aktuátoru a senzoriky s pomocnými elektronickými obvody. Model kromě konstrukce a fyzikálního modelu tří nádrží, tvoří také senzory pro kontinuální měření hladiny, čerpadlo s řídicí elektronikou pro ovládání přítoku do soustavy a řídicí platforma. Jako vodné řešení jsem zvolil hardwarové komponenty, které v dalších podkapitolách podrobně rozepíši i s navrženou řídicí elektronikou. Na následujícím obrázku je obecně zobrazená struktura komponenty fyzikálního modelu, kde je vidět i funkční propojení mezi jednotlivými komponenty.



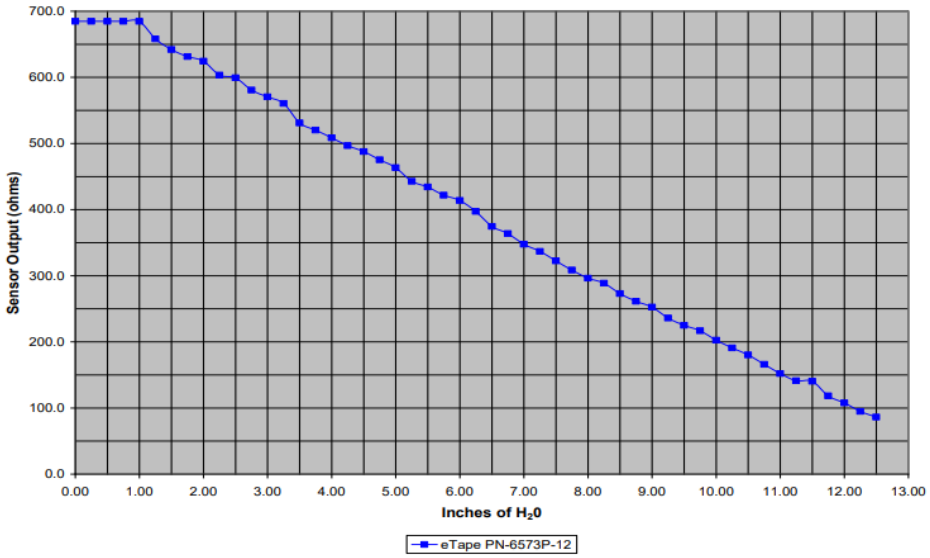
Obrázek 11: *Blokové schéma reálné soustavy*

4.1.1 Hardwarové komponenty

4.1.1.1 Návrh senzorické části

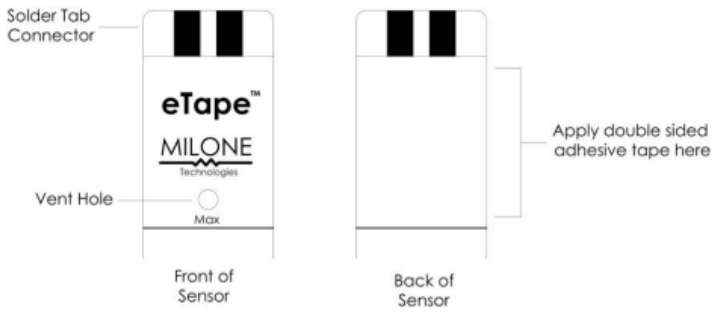
Senzor pro kontinuální měření hladiny Milone eTape, je inovativní polovodičový senzor, který místo pohybu mechanických plováků, využívá tištěnou elektroniku. Obálka eTape je stlačena hydrostatickým tlakem tekutiny, do které je ponořena, což má za následek změnu odporu, která odpovídá vzdálenosti od horní části senzoru k povrchu tekutiny. Senzor si lze představit jako termistor NTC, kde místo ovlivňování odporu změnou teploty, ovlivňujeme odpor změnou hladiny kapaliny.

Pokud hladina kapaliny stoupá, měřený odpor klesá a naopak. Výstupní charakteristika v závislosti na měření výšky je zobrazena na obrázku dole. Pracovní rozsah pro zvolený senzor délky $300\,mm$, který jsem použil k realizaci je $400 - 2000\,\Omega \pm 20\,\%$.



Obrázek 12: *Výstupní charakteristika senzoru eTape [17]*

Aby snímač správně pracoval, nesmí se ohýbat svisle ani podélně. Na horní zadní části senzoru může být nanesena oboustranná lepící páska pro připevnění senzoru k vnitřní stěně měřené nádobky, avšak lepící páska se nemůže nanášet na jinou část z toho důvodu, že není jinak zaručena správná funkčnost. Snímač se nesmí ponořit nad hranici „Max“, aby nedošlo k ponoření odvzdušňovacího otvoru. Tento odvzdušňovací otvor (viz. obrázek) umožňuje senzoru vyrovnávat atmosférický tlak.



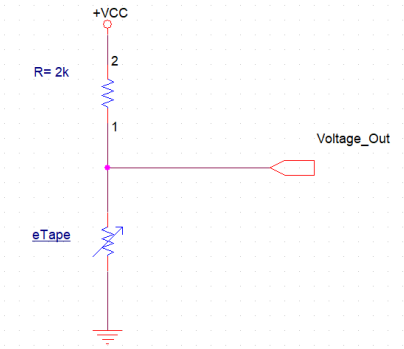
Obrázek 13: *Odvětrávací otvor senzoru eTape [17]*

Pro vyhodnocení naměřených surových hodnot senzorů lze použít více možností dle souvisejícího manuálu. Vyhodnocovací elektronický obvod lze realizovat například jako napěťový dělič, kde převedeme proměnlivé hodnoty odporu snímače na napětí a poté analyzujeme data pomocí A/D převodníku v řídicí jednotce, Nebo lze použít zapojení pomocí invertujícího operačního zesilovače s virtuální nulou. Ten také převádí proměnlivé hodnoty odporu senzoru na napětí, ale vyhodnocení

je o dost přesněji než u napěťového děliče. Tyto dvě varianty jsem navrhl jako možné řešení, proto uvedu obě dvě v následujícím popisu.

Vyhodnocení napěťovým děličem

První návrh využívá jednoduchý zapojení obvodu napěťového děliče k převodu proměnlivého odporu snímače na napětí. Zapojení je znázorněné na obrázku dole. Použitím napěťového děliče získáme výstupní napětí, které úměrné vstupnímu napětí.



Obrázek 14: *Obecné zapojení s napěťovým děličem*

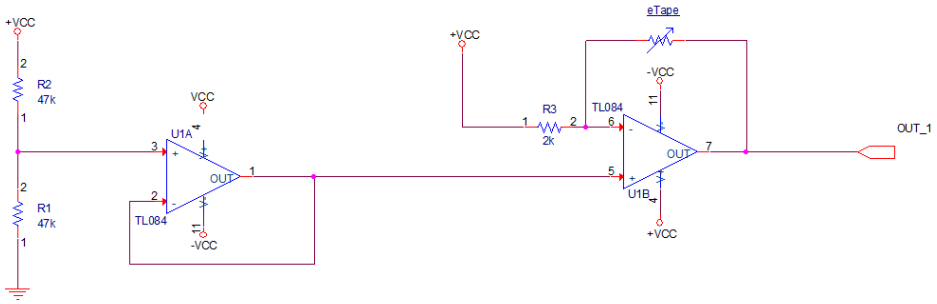
Jak již jsem zmiňoval senzor má nižší odpor při vyšších hladinách a vyšší odpor při nižších hladinách, proto maximální digitální hodnota odpovídá prázdné nádobce a minimální digitální hodnota odpovídá plné nádobce.

$$U_{out} = \frac{U_{cc}.R_{eTape}}{R+R_{eTape}} \tag{35}$$

Vyhodnocení s použitím invertujícího operačního zesilovače

Pro tento případ se vyžaduje použití invertujícího operačního zesilovače, který obecně vyžaduje symetrický zdroj napájení (kladně i záporné napětí). Pro použití tohoto případu v obvodech se zdrojem kladného napětí je třeba použít virtuální zemnící obvod pro vytvoření virtuální nuly. Stejně jako u napěťového děliče je odpor nižší při vyšších hladinách a naopak. Výstupní napětí je dáno dle vztahu:

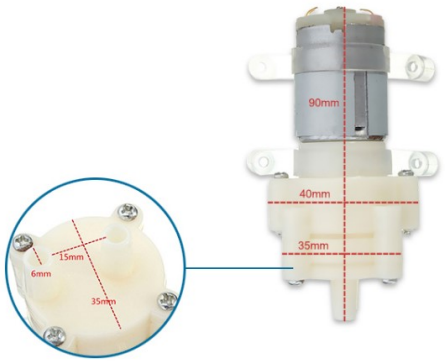
$$U_{out} \approx U_{cc} * \frac{R_{eTape}}{2000} \tag{36}$$



Obrázek 15: *Schéma zapojení s invertujícím OZ pro vyhodnocení napětí*

4.1.1.2 Návrh aktuátoru

Pro napouštění vody do soustavy jsem se rozhodl použít membránové čerpadlo, běžně používané v akvaristice. Napájecí napětí čerpadla je 12 V a provozní proud 0,7 A. Rozměry a vzhled čerpadla jsou znázorněny na obrázku.



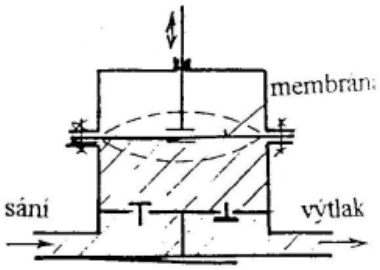
Obrázek 16: *Čerpadlo použité v diplomové práci*

Princip membránového čerpadla:

Hlavním znakem čerpadla je membrána, která se neustále posouvá vpřed a vzad. Tím se v čerpadle vytváří dočasné komory, které nasají a následně vytlačí kapalinu z čerpadla. Proto se o těchto čerpadlech bavíme jako o dvoučinných, objemových. U membránových čerpadel nepřichází čerpaná kapalina do styku s pohyblivými částmi čerpadel, s těsněními a ucpávkami, mohou tedy dopravovat chemicky aktivní či znečištěné kapaliny. Používají se tedy například v chemickém průmyslu, jednoduché membránové čerpadlo bez pístu se také používá jako palivové čerpadlo u spalovacích motorů.

Princip:

Princip činnosti membránových čerpadel je založen na střídavém pohybu pružné části čerpadla zhotovené z pryže, plastu nebo kovu. Membránové čerpadlo tak obsahuje píst a pryžovou membránu, která je připojena na táhlo a třmenem prohýbána nahoru a dolu. Při zdvihání membrány se kapalina nasává do prostoru mezi sacím ventilem a membránou a při klesání se při zavřeném sací ventilu vytlačuje výtlačným hrdlem ven. Pohon čerpadla je zrealizován tak, aby ve finální podobě byl posuvný a pohyboval membránou.[23]

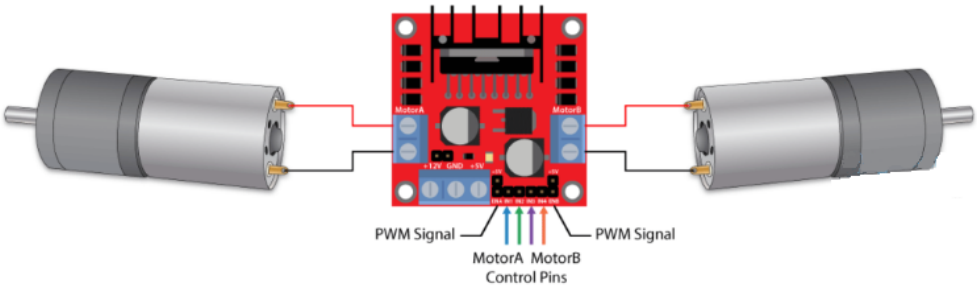


Obrázek 17: *Schéma membránového čerpadla [23]*

Řízení čerpadla pomocí H-můstku

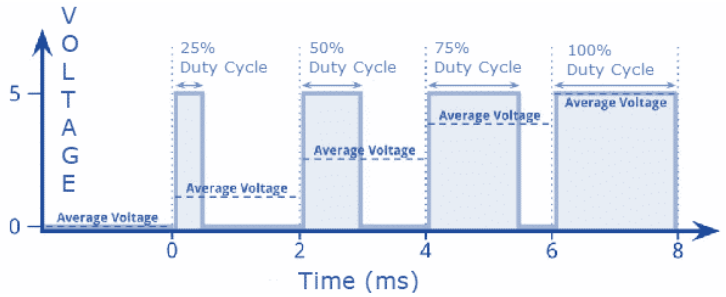
Pro řízení čerpadla jsem použil již hotový modul pro řízení rychlosti i směru stejnosměrných motorů H-můstek L298N. Modul může řídit stejnosměrné motory, které mají napětí mezi $5 - 35\,\mathrm{V}$, se špičkovým proudem do $2\,A$. Modul lze ovládat současně dva motorky, proto má dvě svorkovnice pro řízení motoru *A* a motoru *B*. Další svorkovnice slouží pro připojení vstupního napájení, uzemnění. Nachází se tu i svorka $+5\,V$, která lze použít jako výstupní napájení pro jiné komponenty. Svorka pro řídicí vstupy *Enable A* a *Enable B* se používají k aktivaci a řízení rychlosti motorku. Pokud na těchto kolících je propojka, bude motorek aktivován a pracuje při maximální rychlosti, a pokud odstraníme propojku, můžeme k tomuto kolíku připojit vstup PWM a takto řídit výkon motoru. Pokud tento kolík připojíme k uzemnění, motor bude deaktivován.

Dále vstupní kolíky 1 a 2 jsou použity pro řízení směru otáčení motoru *A* a vstupní kolíky 3 a 4 pro motor *B*. Pomocí těchto kolíků skutečně ovládáme spínače H-můstku uvnitř L298N. Pokud je vstup 1 LOW a vstup 2 je HIGH, motor se bude pohybovat vpřed, a naopak, pokud je vstup 1 HIGH a vstup 2 je LOW, motor se bude pohybovat reverzně. V případě, že jsou oba vstupy stejné, motor LOW nebo HIGH se zastaví. Totéž platí pro vstupy 3 a 4 u řízení pro motor B.[31]



Obrázek 18: *Připojení motoru k L298N [31]*

PWM neboli pulsní šířkovou modulaci, zde využívám jako pulsní regulátor otáček pro řízení stejnosměrného motorku čerpadla. Je to technika, která spočívá v napájení motorku jmenovitým napájecím napětím, které je však připojováno v impulzech po definovaný časový úsek, který je střídán nulovým napájecím napětím. Délka napájecího pulzu potom určuje rychlost otáčení motorku, přičemž se střída může pohybovat od 0 do 100 %. Motor se chová jako kdyby byl napájen napětím o velikosti střední (průměrné) hodnoty, která je dána poměrem doby zapnutí a vypnutí.



Obrázek 19: *Popis PWM modulace [31]*

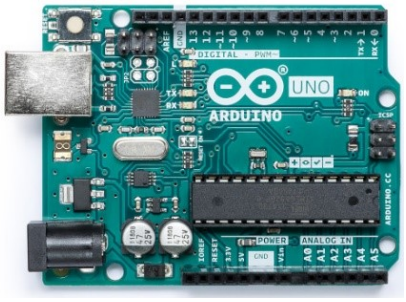
4.1.1.3 Řídicí platforma

Arduino

Arduino je otevřená elektronická platforma založená na jednoduše použitelném hardwaru a softwaru. Je to malý jednodeskový počítač, původně sloužící jako nástroj pro rychlé prototypování, zaměřený pro podporu výuky informatiky pro studenty bez znalostí elektroniky a programování. Je založeno na mikrokontrolerech od firmy Atmel. Obsahují 8bitové mikrokontrolery z rodiny AVR a množství dalších podpůrných obvodů. Arduino používají čipy ATMega8, ATMega168, ATMega328, ATMega1280 a ATMega2560 či dokonce 32-bitový ARM procesor Atmel SAM3X8E. Nelze k němu připojit monitor ani klávesnici či myš, ale lze připojit LED diod, LCD displeje, servomotory, různé senzory atd. V dnešní době se rozšířil pro například i pro vývoj aplikací IoT či 3D tisku. Software pro Arduino je open-source, takže na internetu je k nalezení spousta příspěvků řešení od uživatelů z celého světa.

Je možné projekty realizovat na počítačích Mac, Windows a Linux. Učitelé a studenti ji používají k vytváření levných vědeckých nástrojů, k prokazování principů z oblasti chemie a fyziky nebo jako pomůcku k programování a robotice. Designéři a architekti staví interaktivní prototypy, hudebníci a umělci jej používají pro instalace a experimentování s novými hudebními nástroji.

Základní Arduino Uno použité v diplomové práci obsahuje 14 digitálních vstupních/výstupních pinů (z toho může být 6 použito jako výstupy PWM), 6 analogových vstupů, 16 MHz krystal, připojení pomocí USB, napájecí konektor, ICSP rozhraní a resetovací tlačítko. Obsahuje vše potřebné k provozu mikrokontroléru, jednoduše se připojí k počítači pomocí USB kabelu a může se začít. [18][19]



Obrázek 20: *Arduino UNO R3 [19]*

Raspberry Pi

Raspberry Pi je řada malých jednodeskových počítačů vyvinutých společností Foundation, kterými chtěla firma podporovat výuku základní výpočetní techniky ve školách a v rozvojových zemích. Původní model se stal mnohem populárnějším, než se očekávalo, Raspberry Pi oslovilo i jiný než cílový trh, jako je použití v automatizaci, robotice, či jako multimediální centrum v domácnosti. Nezahrnuje žádná periferní zařízení (například klávesnice a myši) ani pouzdra.

Je to miniaturní počítač, jehož základem je čtyřjádrový procesor Broadcom BCM2837 pracující na frekvenci 1.2 GHz, 1 GB RAM, grafická karta Broadcom VideoCore IV, slot pro microSD karty;

konektory: HDMI, audio výstup, 4x USB, GPIO pro připojení zabezpečovací techniky; LAN, WiFi, Bluetooth a však bez operačního systému. Pro Raspberry Pi byl speciálně vyvinut a optimalizován operační systém Raspbian jenž vychází z operačního systému Debian (jedna z distribucí Linuxu). Operační systém obsahuje sadu základních programů a nástrojů, které zajišťují samotný běh Raspberry Pi.[18]

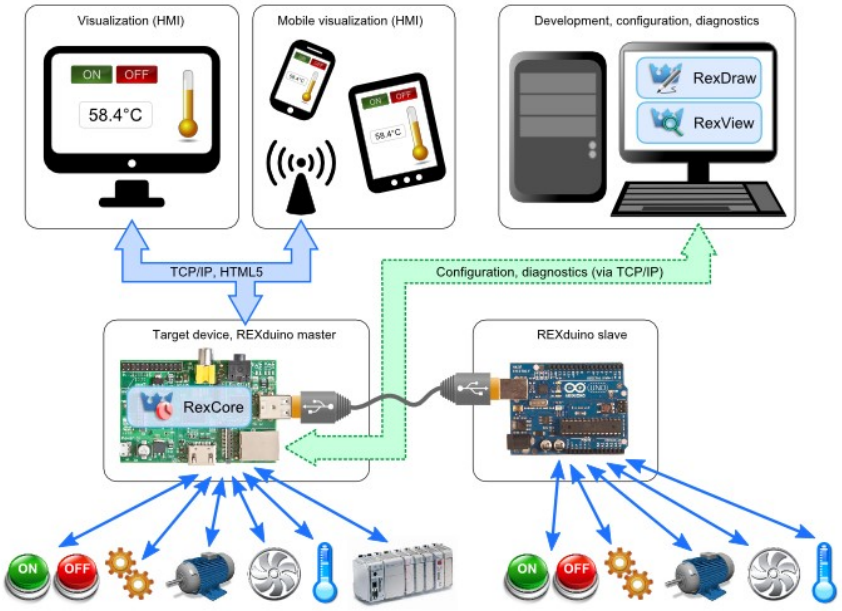
Rozdíl mezi Arduinem a Raspberry Pi je ten, že jsou to různé systémy. Arduino běží na mikrokontroléru, do kterého při naprogramování nahrajeme svůj program a ten běží. Raspberry Pi je mikropočítač, v kterém běží standardní operační systém Linux. Díky tomu s Raspberry Pi můžeme dělat více složitých věcí, ale zároveň máme k dispozici propojovací GPIO piny podobně jako u Arduina, na které můžeme připojit další periferie. Stručně řečeno Raspberry Pi může nahradit Arduino, ale nikoliv naopak.[20]

Obrázek 21: *Raspberry Pi 3 Model B v1.2 [21]*

ARM architektura využívá redukované instrukční sady RISC, což se významně projevuje na nízké spotřebě energie a tyto procesory tak nepotřebují dodatečné chlazení na rozdíl od procesoru stolních počítačů, které pracují s komplexní instrukční sadou CISC. Z těchto důvodů lze Raspberry Pi napájet přes integrovaný port mikro USB zdrojem napájení s napětím $5\,V$ a proudem $1\,A$. Nevýhodou této architektury je nekompatibilita většiny softwaru určených pro stolní počítače, jelikož ten využívá architekturu s instrukční sadou x86.[18][20][21]

Platforma Rexduino

Platforma Rexduino je spojení mikrokontroleru Arduino s minipočítačem Raspberry Pi s pomocí řídicího systému Rexygen. Kombinací těchto zařízení vzniká velmi levná a výkonná platforma pro řízení algoritmů v reálném čase. Komunikace mezi Raspberry Pi a Arduinem je vytvořena skrz USB a stará se o ní řídicí systém Rexygen. Symbióza řídicího systému Rexygen a mikrokontrolérem Arduino, je založena na jednoduchém komunikačním protokolu, fungujícím na principu počítačového modelu Master/Slave. Do zařízení Slave, které je v tomto případě Arduino, je nainstalován program, který zpracovává data přijímaná ze zařízení Master přes USB a vykonává požadované operace, popřípadě odpovídá zpět Masteru, kde Master představuje Raspberry Pi. Zařízení Slave funguje v podstatě jako modul vstupů, výstupů, zpracovávající informace ze senzorů, či ovládání aktuátorů. Na cílovém zařízení Master neboli Raspberry Pi je nainstalováno runtime jádro RexCore, které se stará o provedení algoritmu realizovaného funkčními bloky dle standardu IEC 61131-3, stejně jako u PLC. Algoritmus je naprogramován na PC a nahrán do Raspberry Pi přes TCP/IP, tak je možné i diagnostikovat řízený proces.[18]



Obrázek 22: *Principiální schéma platformy REXduino [22]*

- Zkompilovat sketch REXduino_slave pomocí Arduino IDE a spustit ho na Arduinu
- Nainstalovat jádro řídicího systému Rexygen na Raspberry Pi
- Zapojit Arduino do Raspberry Pi
- Nainstalovat vývojové nástroje řídicího systému Rexygen na PC
- Zkompilovat projekt Rexygen master a nahrát ho do Raspberry Pi přes PC, na kterém běží jádro RexCore.

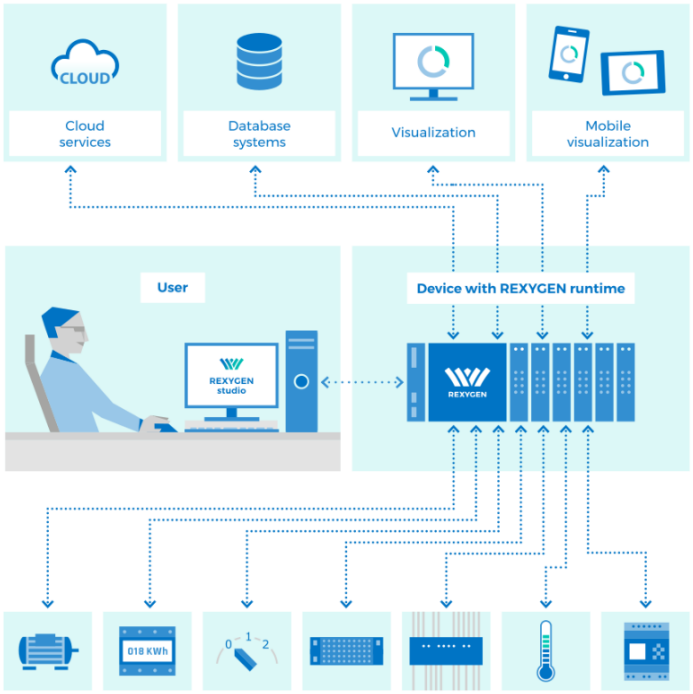
4.1.2 Software pro řízení Rexygen

Rexygen je řídicí systém vyvinutý firmou Rex controls s.r.o. Své využití již našel v různých oblastech automatizace, měření, robotiky, řízení a kontroly procesů a dalších. Uživateli umožňuje vývoj, kompilaci i analýzu projektů.

Jádro runtime systému Rexygen (RexCore) je základním kamenem automatizačního zařízení na kterém je nainstalováno. Běží nad operačním systémem a koordinuje provádění algoritmů. Poskytuje také přístup ke vstupním a výstupním signálům platformy prostřednictvím modulárního systému ovladačů Rexygen I/O. Struktura systému Rexygen je zobrazena na obrázku dole.

Vlastnosti systému Rexygen:

- Grafické programování bez ručního kódování.
- Programování řídicích jednotek na standardním PC nebo notebooku.
- Uživatelské rozhraní pro stolní počítače, tablety a chytré telefony (HMI).
- Široká rodina podporovaných zařízení a vstupně-výstupních jednotek (včetně Raspberry Pi).
- Osvědčené řídicí algoritmy.
- Snadná integrace do podnikové IT infrastruktury (ERP / BMS).
- REST API pro bezproblémovou integraci do řešení Industry 4.0 a IoT



Obrázek 23: *Struktura REXYGEN [30]*

Hlavní komponenty softwaru Rexygen:

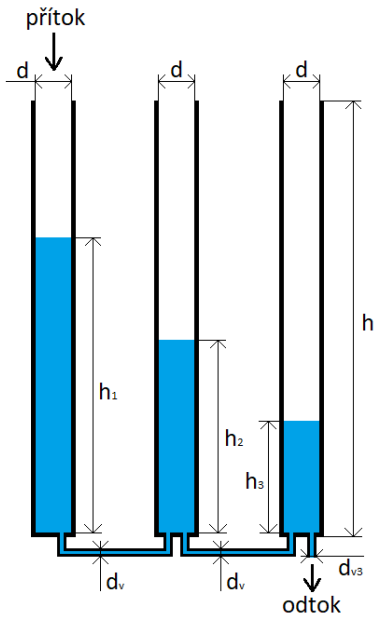
- **Rexygen Studio** – Vývojové prostředí, které běží na standardním PC se systémem Windows. Algoritmy se vytváří pomocí knihovny funkčních bloků (Norma IEC 61131-3), což programovací jazyk odpovídající standardu PLC. Knihovna obsahuje různé logické členy, komparátory, časovače, bloky zpracovávající signály, PID regulátory, až po bloky archivace dat. Po spuštění lze algoritmus sledovat v reálném čase. Můžem se připojit přes místní síť nebo přes internet.
- **Rexygen HMI Designer** – Nástroj pro navrhování grafického prostředí pro navržené algoritmy. Je implementován do jiného vývojového nástroje Inkscape, který pracuje s vektorovou grafikou. HMI designer disponuje knihovnou již vytvořených grafických komponent, které lze snadno použít pro tvorbu naší vizualizace, to ale nebrání vytvořit si komponenty vlastní.
- **Kompilátor Rexygen** – Pro uživatele neviditelný, vyvolán je vývojovým prostředím Rexygen Studio. Převádí algoritmy do strojového kódu.
- **RexCore** – Runtime jádro Rexygen běží na cílovém zařízení v našem případě Raspberry Pi. Po nainstalování běží na pozadí a nijak nezasahuje do běhu operačního systému. Provádí multitasking, dále obsahuje integrovaný webový server poskytující uživatelské rozhraní HMI a REST API.
- **Diagnostika Rexygen** – Nástroj pro diagnostikování runtime jádra, pro provedení do provozu či diagnostiku řídicích algoritmů. Je součástí Rexygen Studia.[30]

5 Návrh a realizace nelineárního prediktivního řízení

Pro navržení řídicího algoritmu je prvotní znalost modelu vycházejícího z popisu reálné soustavy tří nádrží. Dalším krokem je zvolení vhodně kriteriální funkce, jejíž lokální extrém bude předmětem optimalizace a bude odpovídat řešenému problému. Poté je třeba navrhnout penalizační váhové matice Q, R, P a také celkový počet kroků s predikčním horizontem N.

5.1 Matematický model

Matematický model vychází ze známé problematiky řízení systému tří nádrží. Model se skládá ze tří kaskádně propojených nádrží s přítokem do první nádrže a odtokem třetí poslední nádrže (obrázek níže). NMPC regulátor se navrhuje tak, aby ve třetí nádrži byla dosažena zadaná výška hladiny. Jako počáteční stav je třeba brát v úvahu nulovou výšku hladiny ve všech tří nádrží. Nádrže včetně jejich propojení mají stejné parametry.



Obrázek 24: *Model tří kaskádně propojených nádrží*

Každá z nádob lze popsat následující diferenciální rovnicí:

$$\mathrm{S}\frac{dh(t)}{dt} = Q(t) - \mathrm{S_v} \cdot v(t) \tag{37}$$

kde

S – Plocha nádoby $[m^2]$

S_v – Plocha odtokového otvoru z první a druhé nádoby $[m^2]$

S_{v3} – Plocha odtokového otvoru ze třetí nádoby $[m^2]$

$Q_{(t)}$ – Přítok do nádoby $\left[\frac{\mathrm{m}^3}{s}\right]$

$v_{(t)}$ – výtoková rychlost z nádoby $\left[\frac{\mathrm{m}}{s}\right]$

$h_{(t)}$ – hladina kapaliny v nádobě $[m]$

Výtoková rychlost je popsaná Torriceliho vzorcem:

$$v(t) = \sqrt{2gh(t)} \tag{38}$$

Při kaskádním propojení více nádob se vztah upraví s ohledem na výšku hladiny kapaliny v sousední nádobě:

$$v_{12}(t) = signh\big(h_1(t) - h_2(t)\big)\sqrt{2g|h_1(t) - h_2(t)|} \tag{39}$$

Znaménková funkce signum zajišťuje, že při záporném rozdílu hladin h_1 a h_2 se změní i směr rychlosti. Absolutní hodnota rozdílu hladin pod odmocninou zamezuje překročení definičního oboru odmocniny. Ale jelikož víme, že rozdíl hladin h_1 - h_2 je vždy kladný, protože počáteční hladiny v jednotlivých nádobách mají sestupnou tendenci (směr rychlosti toku kapaliny bude vždy od prvního k poslednímu). Rovnici zapíšeme ve zjednodušeném tvaru:

$$v(t) = \sqrt{2g(h_1(t) - h_2(t))} \tag{40}$$

Pro všechny tři nádoby pak dostáváme soustavu tří nelineárních diferenciálních rovnic popisující změnu výšky kapaliny v každé nádobě.

$$\frac{dh_1(t)}{dt} = \frac{Q(t)}{S_1} - \frac{S_{v12}}{S_1}\sqrt{2g(h_1(t) - h_2(t))} \tag{41}$$

$$\frac{dh_2(t)}{dt} = \frac{S_{v12}}{S_2}\sqrt{2g(h_1(t) - h_2(t))} - \frac{S_{v23}}{S_2}\sqrt{2g(h_2(t) - h_3(t))} \tag{42}$$

$$\frac{dh_3(t)}{dt} = \frac{S_{v23}}{S_2}\sqrt{2g(h_2(t) - h_3(t))} - \frac{S_{v3}}{S_3}\sqrt{2gh_3(t)} \tag{43}$$

Rovnice platí za předpokladu o shodných rozměrech tří nádob a parametrech jednotlivých propojení.[16]

Transformace na metodu RungeKutta 4. řádu

Prediktivní řízení je ve své podstatě založeno na použití diskrétních, respektive vzorkovaných modelech procesů, a proto odvození příslušných řídicích algoritmů je realizováno hlavně v diskrétní oblasti. Proto použijeme metodu RungeKutta 4. řádu:

Určíme první pomocný bod pomocí Eulerovy metody s krokem $\frac{1}{2}h$:

$$k_{11} = -a_1\sqrt{x_{01} - x_{02}} + \frac{Q}{S} \tag{44}$$

$$k_{12} = a_1\sqrt{x_{01} - x_{02}} - a_2\sqrt{x_{02} - x_{03}} \tag{45}$$

$$k_{13} = a_2\sqrt{x_{02} - x_{03}} - a_3\sqrt{x_{03}} \tag{46}$$

Kde

$$a_1 = \frac{S_v}{S}\sqrt{2.g} \tag{47}$$

$$a_2 = \frac{S_v}{S}\sqrt{2.g} \tag{48}$$

$$a_3 = \frac{S_{v3}}{S}\sqrt{2.g} \tag{49}$$

Pomocný bod:

$$x_{1p} = x_{01} + \frac{k_{11}.h}{2} \tag{50}$$

$$x_{2p} = x_{02} + \frac{k_{12}.h}{2} \tag{51}$$

$$x_{3p} = x_{03} + \frac{k_{13}.h}{2} \tag{52}$$

Určíme druhý pomocný bod pomocí derivace v prvním bodě s krokem $\frac{1}{2}h$:

$$k_{21} = -a_1\sqrt{x_{1p} - x_{2p}} + \frac{Q}{S} \tag{53}$$

$$k_{22} = a_1\sqrt{x_{1p} - x_{2p}} - a_2\sqrt{x_{2p} - x_{3p}} \tag{54}$$

$$k_{23} = a_2\sqrt{x_{2p} - x_{3p}} - a_3\sqrt{x_{3p}} \tag{55}$$

Pomocný bod:

$$x_{1q} = x_{01} + \frac{k_{21}.h}{2} \tag{56}$$

$$x_{2q} = x_{02} + \frac{k_{22}.h}{2} \tag{57}$$

$$x_{3q} = x_{03} + \frac{k_{23}.h}{2} \tag{58}$$

Určíme třetí pomocný bod pomocí derivace v druhém bodě s krokem h:

$$k_{31} = -a_1\sqrt{x_{1q} - x_{2q}} + \frac{Q}{S} \tag{59}$$

$$k_{32} = a_1\sqrt{x_{1q} - x_{2q}} - a_2\sqrt{x_{2q} - x_{3q}} \tag{60}$$

$$k_{33} = a_2\sqrt{x_{2q} - x_{3q}} - a_3\sqrt{x_{3q}} \tag{61}$$

Pomocný bod:

$$x_{1e} = x_{01} + \frac{k_{31}.h}{2} \tag{62}$$

$$x_{2e} = x_{02} + \frac{k_{32}.h}{2} \tag{63}$$

$$x_{3e} = x_{03} + \frac{k_{33}.h}{2} \tag{64}$$

Provedeme derivaci ve třetím bodě:

$$k_{31} = -a_1\sqrt{x_{1e} - x_{2e}} + \frac{Q}{S} \tag{65}$$

$$k_{32} = a_1\sqrt{x_{1e} - x_{2e}} - a_2\sqrt{x_{2e} - x_{3e}} \tag{66}$$

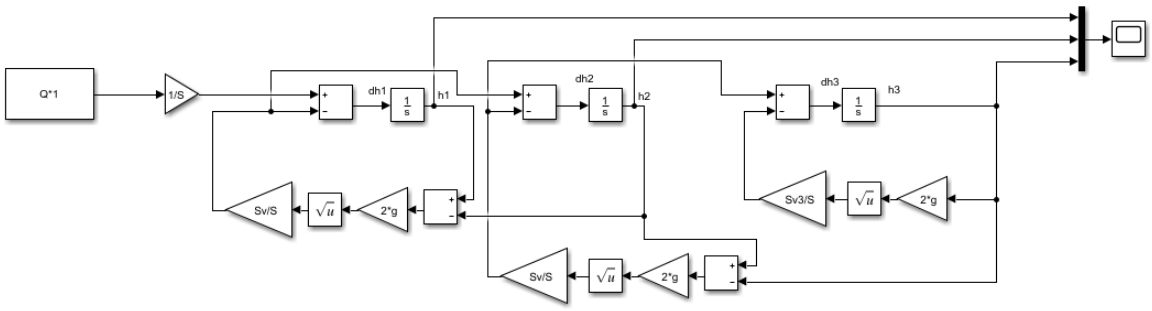
$$k_{33} = a_2\sqrt{x_{2e} - x_{3e}} - a_3\sqrt{x_{3e}} \tag{67}$$

Vypočítáme vážený průměr derivací ve výchozím bodě a ve třech pomocných:

$$x_1 = x_{01} + \frac{h(k_{11}+2k_{21}+2k_{31}+k_{41})}{6} \tag{68}$$

$$x_2 = x_{02} + \frac{h(k_{12}+2k_{22}+2k_{32}+k_{42})}{6} \tag{69}$$

$$x_3 = x_{03} + \frac{h(k_{13}+2k_{23}+2k_{33}+k_{43})}{6} \tag{70}$$



Obrázek 25: *Zobrazení modelu soustavy pomocí Simulinku*

5.2 Návrh prediktivního regulátoru

Regulátor "controller.m" je realizován v prostředí Matlab skriptovacím programovacím jazykem. Nastavený celkový počet kroků cpk byl stanoven na 1000. To je dostatečně dlouhá doba na prezentování výsledků navrženého řídicího algoritmu. Perioda vzorkování T_s je stanovena na jednu sekundu, tedy každou vteřinu proběhne cyklus algoritmu provádějící výpočet optimalizovaných budoucích akčních zásahů. Jsou stanovena omezení v oblasti přítoku do oblasti. Omezení jsou zahrnuta do optimalizačního výpočtu nelineární funkce, který provádí funkce *fmincon*. Jednotlivé oblasti jsou dále popsány v podkapitolách.

Účelová funkce

Funkce, která při určitých hodnotách parametrů poskytne svou funkční hodnotu k vyhodnocení neboli udává kvalitu řešení optimalizace. Pomocí změn parametrů je vyhledána taková hodnota účelové funkce, která bude z daného hlediska optimální. Účelová funkce je realizována z obecného znění:

$$J = \min_{u}\left[\left(x_N^T - x_{ref}\right).P.\left(x_n - x_{ref}\right) + \sum_{k=0}^{N-1}\left\{\left(x_k^T - x_{ref}\right).Q.\left(x_k - x_{ref}\right) + u_k^T.R.u_k\right\}\right] \quad (71)$$

kde

Q, R, P – váhové matice

x_{ref} – reference

Penalizační matice

Pro určení penalizační matice v případě NMPC neexistuje předepsaný postup. Teoreticky každá volba matic Q a R (splňující požadavek na pozitivní semidefinitnost resp. definitnost) vede na optimální regulátor, nicméně volba konkrétních hodnot je provedena expertně s ohledem na požadavky na regulační pochod.

Zvolené hodnoty pro matice:

$$Q = \begin{vmatrix} 5.10^{-5} \\ 5.10^{-4} \\ 0 \end{vmatrix} \quad (72)$$

$$R = |2| \quad (73)$$

$$P = \begin{vmatrix} 0{,}00275 \\ 0{,}01 \\ 0 \end{vmatrix} \quad (74)$$

Horizont Predikce

Hodnota horizontu N predikce byla opět zvolena expertní volbou, návrh počítá s délkou 30 kroků. Při zvolené periodě vzorkování je to dostatečná doba na reakci při změně referenční trajektorie.

$$N = 30\,s \tag{75}$$

Fmincon

Pro optimalizační výpočet je použita funkce *fmincon*, která je součástí optimalizačního toolboxu v Matlabu. Funkce hledá minimum omezené nelineárního funkce více proměnných. Mezi její jednoznačné klady patří možnost zvolit mezi několika přístupy k nelineární optimalizaci a obecně velké množství nastavení.

Omezení

Omezení regulátoru vychází z omezení průtoku čerpadla, kdy volíme dolní mez jako nulovou a horní mez jako maximální možný řízený přítok. Bavíme se o omezení akčního zásahu:

$$0 \leq u(k) \leq 2{,}2.\,10^{-5} \tag{76}$$

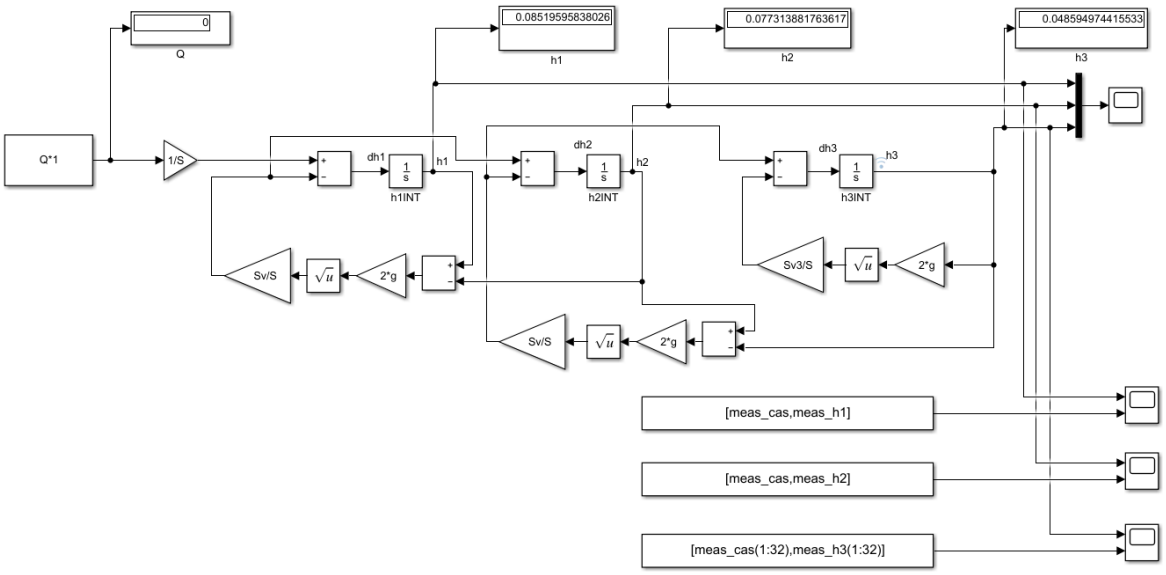
5.3 Identifikace soustavy

Identifikaci parametrů systému provádíme z důvodu nesouladu reálné soustavy s matematickým modelem, s nímž pracujeme při simulaci. Z důvodu vysoké citlivosti dynamických vlastností soustavy zejména na průřezu odtokové trubice ze soustavy. Tyto parametry a výpočet nové průtokové charakteristik pro řízení čerpadla realizující přítok do soustavy řešíme v následující části diplomové práce. Součástí elektronické přílohy je video s postupem sloužící jako návod pro provedení identifikace.

Identifikace z dynamických odezev

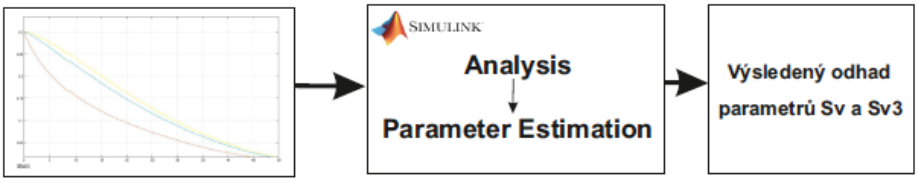
Identifikace z dynamických odezev proběhla tak, že čerpadlo bylo ponecháno ve vypnutém stavu (PWM=0). Tím dosáhneme toho, aby nám prozatím neznámá charakteristika neovlivňovala výsledek pokusu. Následně bylo potřeba změřit v čase, po dostatečně dlouhou dobu, charakteristiky vycházející z odezvy na počáteční stav. Při pokusu se externě naplnily nádoby fyzikálního modelu na hodnotu $h_1 = h_2 = h_3 = 0{,}3\,m$, z toho důvodu bylo důležité uzavřít odtok soustavy. Posledním krokem bylo uvolnění odtoku soustavy a odečítáním hladiny pomocí měrných rysek senzorů zobrazující výšku sloupce kapaliny. Hodnoty sloupce vody se odečítaly po uplynutí jedné sekundy. Výsledky měření v tabulce, jsou součástí elektronické přílohy.

S odečtenými výsledky měření a s parametry soustavy se vytvořil skript v Matabu (součástí elektronické přílohy) „nadoby_podklad.m“, který nahraje data do Workspace. Dále se spustí model soustavy „model.slx“ vymodelovaný z matematického popisu soustavy, na kterém lze pokus simulovat.



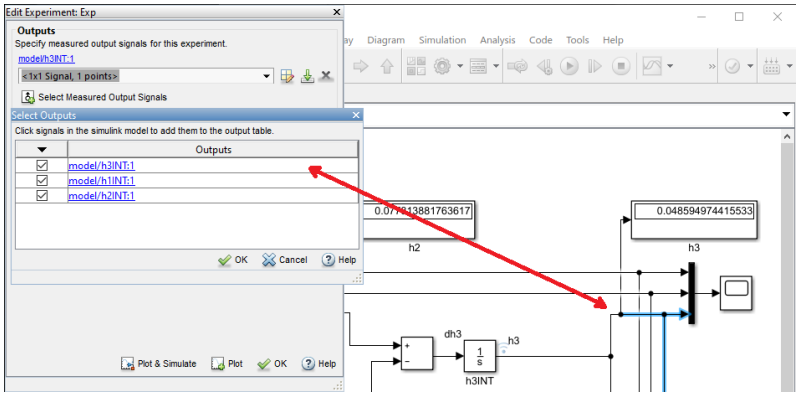
Obrázek 26: *Matematický model použitý pro identifikaci parametrů*

Pro simulaci byl třeba zvolit čas simulace na $32\ s$. Je to proto, že při zhlédnutí výsledků pokusu byly po uplynutí této doby hodnoty ve třetí nádrži neidentifikované. Stalo se tak proto, že výsledky odečítání byly relevantní a špatně se odečítaly skrze objímku držící nádobku s konstrukcí modelu. Po spuštění simulace lze porovnat simulované výsledky od naměřených pro jednotlivé nádoby blokem *Scope*.



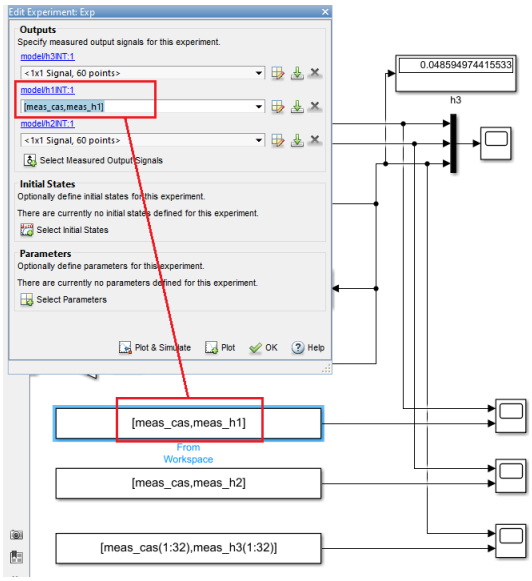
Obrázek 27: *Blokové schéma principu odhadu parametrů*

Stručný popis, čeho jsme chtěli v následujících krocích dosáhnout je zobrazen blokovým schématem na obrázku nahoře, v podstatě z naměřených dat chceme promocí specializovaného nástroje Matlabu *Parameter estimation* provést identifikaci parametrů. Základem pro jeho použití je mít nachystané údaje ve Workspace, který chceme ladit. Nástroj *Parameter estimation* najdeme pod kategorií *Analysis*. Vytvořili jsme nový experiment a otevřelo se nám okno s možnostmi, kde jsme vybrali zvolené měřící signály z modelu soustavy (obrázek níže), což jsou hladiny nádrží. Signály lze vybrat tak, že je označíme přímo v modelu Simulinku.



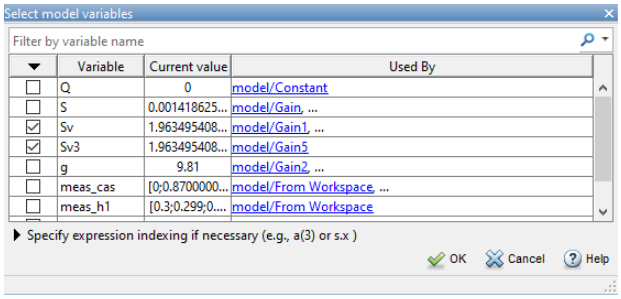
Obrázek 28: *Výběr signálů*

Do řádků, který vzniknou pod názvy signálů po výběru všech tří hladin, se zkopírovaly měřená data. Musí se dávat pozor a přiřadit měřená data ke správnému signálu (viz. obrázek 29).



Obrázek 29: *Správné přiřazení měřených dat k signálům*

Dále *Initial states* nebylo třeba nastavovat, tyto údaje si nástroj bere z integrátorů v Simulinku. V sekci *Parameters* se vybraly parametry, které chceme ladit, v našem případě S_v a S_{v3} a celý nastavení potvrdíme tlačítkem *OK*.



Obrázek 30: *Volba parametrů*

Nyní stačilo spustit vyhledávání odhadů tlačítkem *Estimation* a nástroj začal vykreslovat grafy s vyhledávanými odhady. Po dokončení je třeba výsledky uložit tlačítkem *Save iteration* (viz. Obrázek 31). Získané parametry se uložily do Workspace a výsledek s nově získanými parametry je již možné zkontrolovat v Simulinku pomocí *Scope* pro jednotlivé hladiny. Je vidět, že se průběhy srovnaly a to znamená, že získané výsledky můžeme použít do další fáze identifikace. Proto je ale třeba převést parametry S_v a S_{v3} na d_v a d_{v3}, k tomu použijeme vztahy:

$$d_v = \sqrt{\frac{4.S_v}{\pi}} \tag{77}$$

$$d_{v3} = \sqrt{\frac{4.S_{v3}}{\pi}} \tag{78}$$

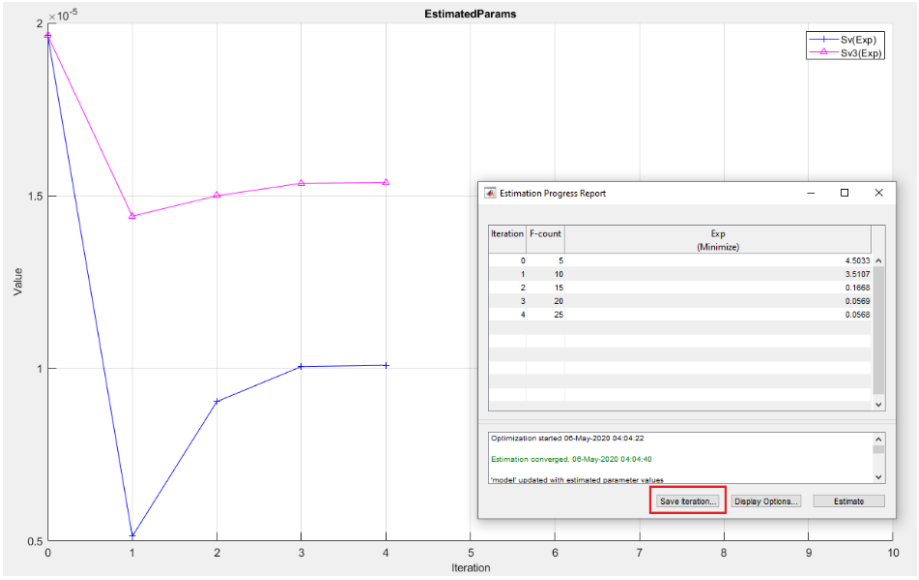
Vycházející z obecného vztahu:

$$S = \frac{\pi}{4}.{d_v}^2 \tag{79}$$

Z toho vyplývá, že výsledný parametry, které použijeme dále v identifikaci jsou:

$$d_{v3} = 0{,}0044 \tag{80}$$

$$d_v = 0{,}0036 \tag{81}$$



Obrázek 31: *Výsledek vyhledávání parametrů a uložení do Workspace*

Stanovení průtokové charakteristiky:

Jelikož ke stanovení d_v a d_{v3} nebylo potřeba průtoku, protože čerpadlo bylo vypnuté. Můžeme nyní použít nalezené parametry pro určení průtokové charakteristiky. V tomto případě vycházíme z naměřených hodnot ustálených stavů na fyzikálním modelu pro tři různé hodnoty PWM a to:

PWM = 182:

$$h_{1ust} = 25\,cm\,; h_{2ust} = 14\,cm\,;\; h_{3ust} = 2\,cm \tag{82}$$

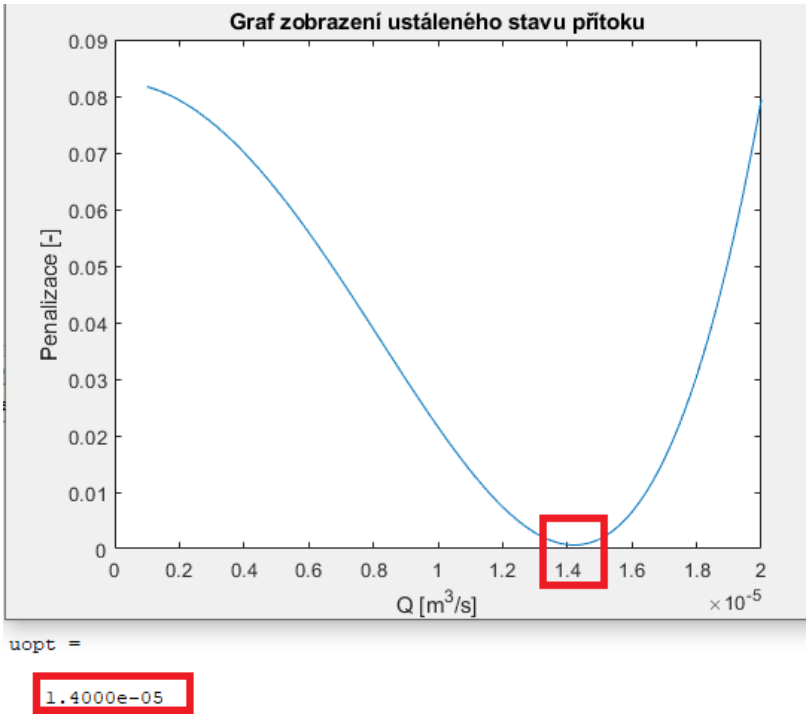
PWM = 170:

$$h_{1ust} = 20\,cm\,; h_{2ust} = 11\,cm\,;\; h_{3ust} = 0{,}7\,cm \tag{83}$$

PWM = 136:

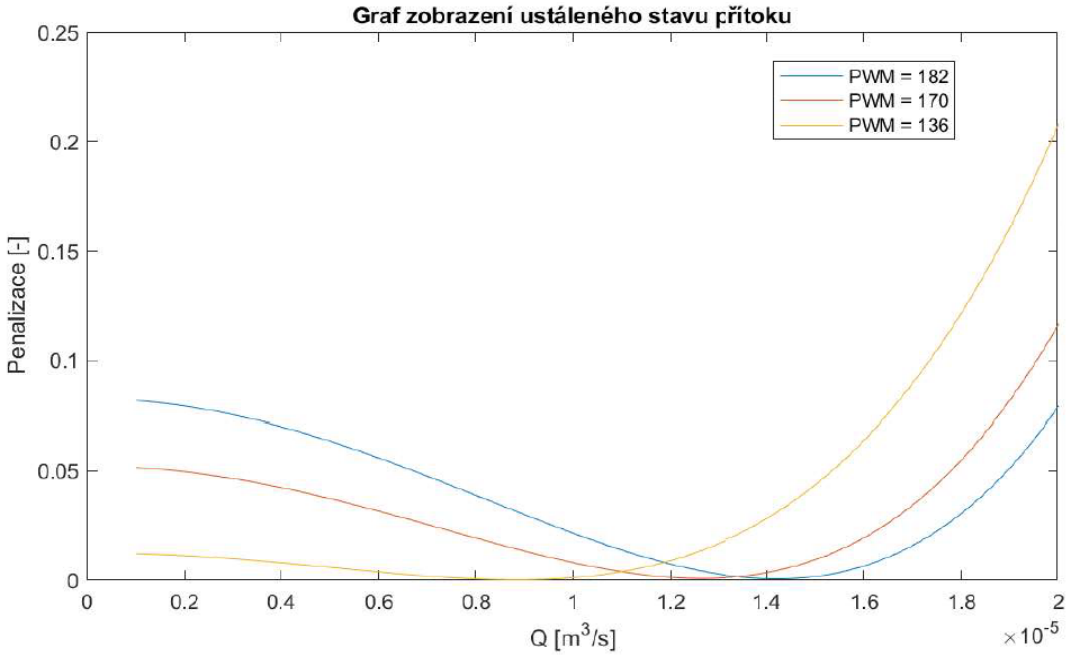
$$h_{1ust} = 10\,cm\,;\; h_{2ust} = 5\,cm\,;\; h_{3ust} = 0{,}3\,cm \tag{84}$$

Opět byl vytvořený skript v Matlbu „nalezt_ustaleny_pritok.m“, který vyhledá ustálený stav pomocí funkce *fminsearch*. Funkce podobná funkci *fmincon* s tím, že *fmincon* zahrnuje do výpočtu omezení. Také ale vykresluje graf, který zobrazuje minimální odchylku ustáleného stavu přítoku a z něhož lze vyčíst a také určit hodnotu minimální odchylky ustáleného stavu přítoku, tím způsobem, že nejmenší hodnota penalizace (y-osa) představuje nejlepší řešení situace. S porovnáním výsledné hodnoty funkce *fminsearch* je vidět shoda. Názorná ukázka je řešení varianty s PWM = 182.



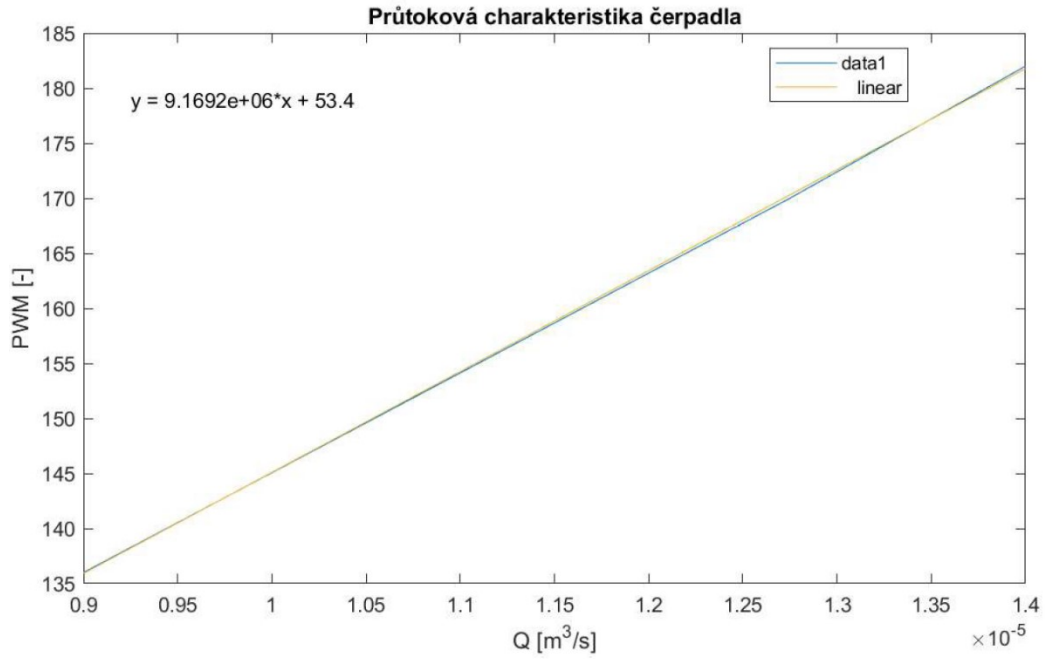
Obrázek 32: *Porovnání hodnot vyčtených z grafu a výsledku funkce fminsearch*

Pro ukázku přikládám rozdíl mezi ustálenými průtoky:



Obrázek 33: *Porovnání ustálených průtoků pro zvolené hodnoty PWM*

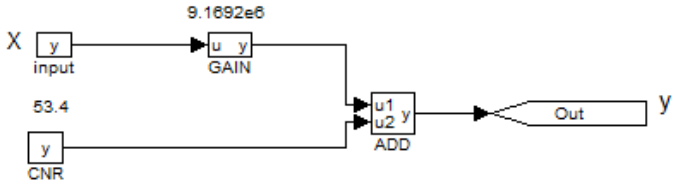
Jakmile byly získány hodnoty ustálených průtoků, použily se v dalším skriptu „prutok_chra_cerpadla.m“, kde se vykreslí průtoková charakteristika čerpadla, na níž je vidět, že je lineární.



Obrázek 34: *Průtoková charakteristika čerpadla s vygenerovanou funkcí*

Pomocí nástroje *Basic Fitting* se určil tvar rovnice. Který se použije do řídicího algoritmu v Rexygenu ve formě funkčních programovatelných bloků. Tato část kódu bude ve finálním řídicím algoritmu před vstupem pro řízení PWM výstupu na Arduinu.

$$y = 9{,}1692.10^6 x + 53{,}4 \tag{85}$$



Obrázek 35: *Rovnice vymodelovaná v Rexygenu*

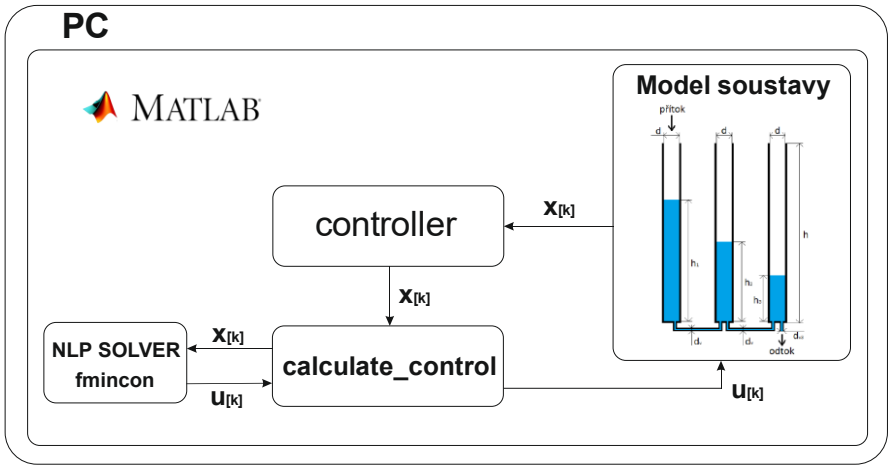
6 Ověření navržených řídicích algoritmů v simulaci.

Myšlenkou simulace je co nejlépe nahradit zkoumaný reálný systém pomocí matematického modelu neboli simulátoru s tím, že se simulátorem experimentujeme pro získání co nejvíce informace o původním zkoumaném reálném systému. V diplomové práci simulaci využíváme pro zjištění fungování navrženého regulátoru, před samotným nasazením na fyzikální soustavu. V určitých fází vývoje řídicího systému je složité testovat návrh přímo na reálně soustavě. Před zavedením algoritmů na cílové zařízení využíváme několik typů simulací, kterými lze odladit řadu problémů.[25]

- **Model in the loop**: Matematické model je simulován s navrženým řídicím systémem v simulačním prostředí na jednom počítači.
- **Software in the loop**: Matematické model je simulován v reálném čase s navrženým řídicím systémem v simulačním prostředí na jednom počítači.
- **Processor in the loop**: Matematický model je simulován v reálném čase, navržený řídicí systém je provozován na cílové hardwarové platformě.
- **Hardware in the loop:** Podobná simulaci PIL. Navržený řídicí systém je provozován na cílové hardwarové platformě, ale nyní již s použitím vstupů a výstupů platformy, případně emulace čidel a akčních členů.

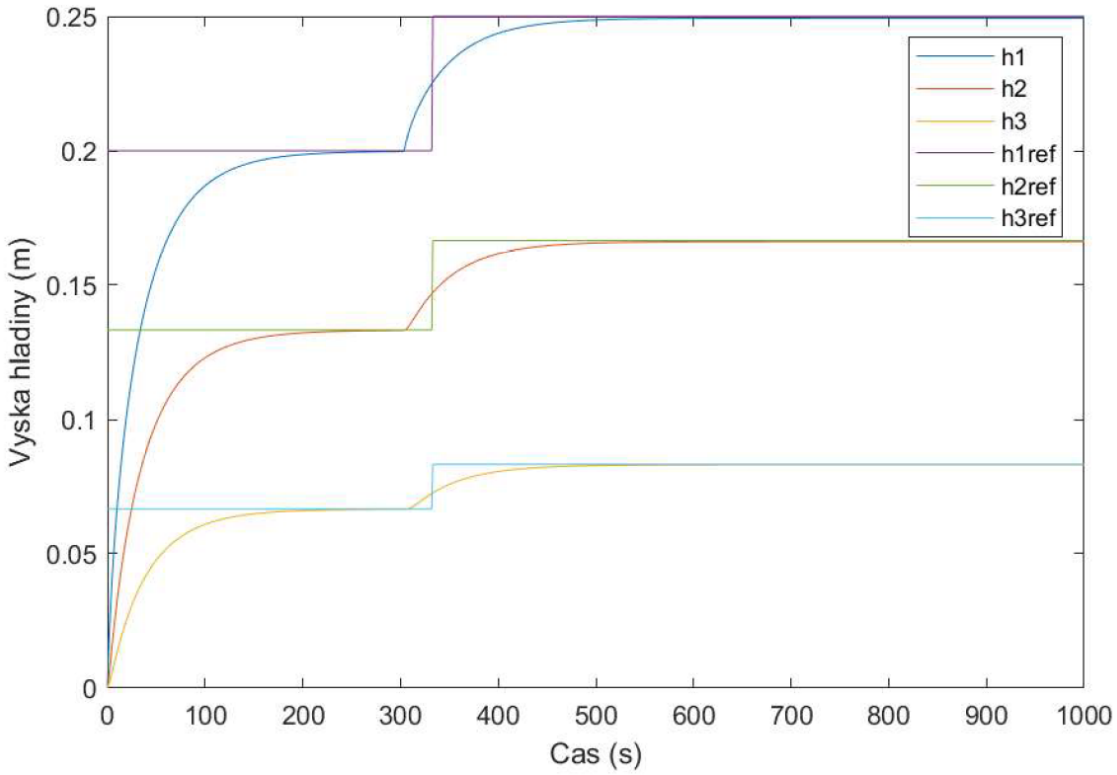
6.1 Simulace pomocí MIL

Simulace MIL vychází ze samotného návrhu prediktivního řízení pro nelineární model. Poté co byl navržen matematický model soustavy s návrhem řídicího algoritmu regulátoru, je potřeba ověřit, zda regulátor je schopný řídit zařízení. Simulace je tak vytvořena pomocí Matlabu spolu s regulátorem na jednom počítači. Principiální schéma je zobrazeno na obrázku níže. Je navržena tak, že je matematický model zapsán ve skriptovacím programovacím jazyce v Matlabu vycházejícího z jazyka Fortran a k ní je naprogramovaný navržený regulátor starající se o predikci a aplikování akčních zásahů na simulovanou řízenou soustavu. Řídicí algoritmus je, jak již bylo zmíněno zapsán ve formě "source-code" a je součástí elektronické přílohy.



Obrázek 36: *Principiální schéma simulace MIL*

Výstup ze simulace je znázorněn na obrázku dole, kde jsou vykresleny hladiny nádrží reagující na změnu reference. Jak je vidět z grafu reference se v čase přibližně 333 s mění na vyšší hodnotu při čemž akční zásah reaguje o nastavený horizont predikce $N = 30$ v předstihu a snaží se dosáhnout co nejblíže k referenční trajektorii.

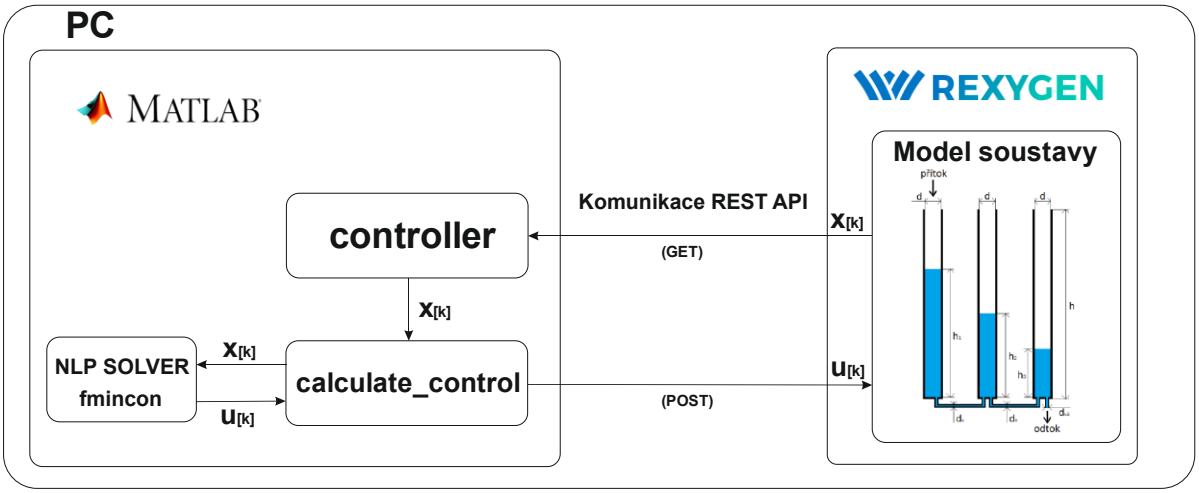


Obrázek 37: *Výstupní charakteristika ze simulace MIL*

Simulace MIL testuje řízení na simulovaném modelu soustavy dle předpokladů. Díky experimentování se simulací jsme schopny odladit parametry a správné nastavit penalizační matice pro použití v další fázi simulací.

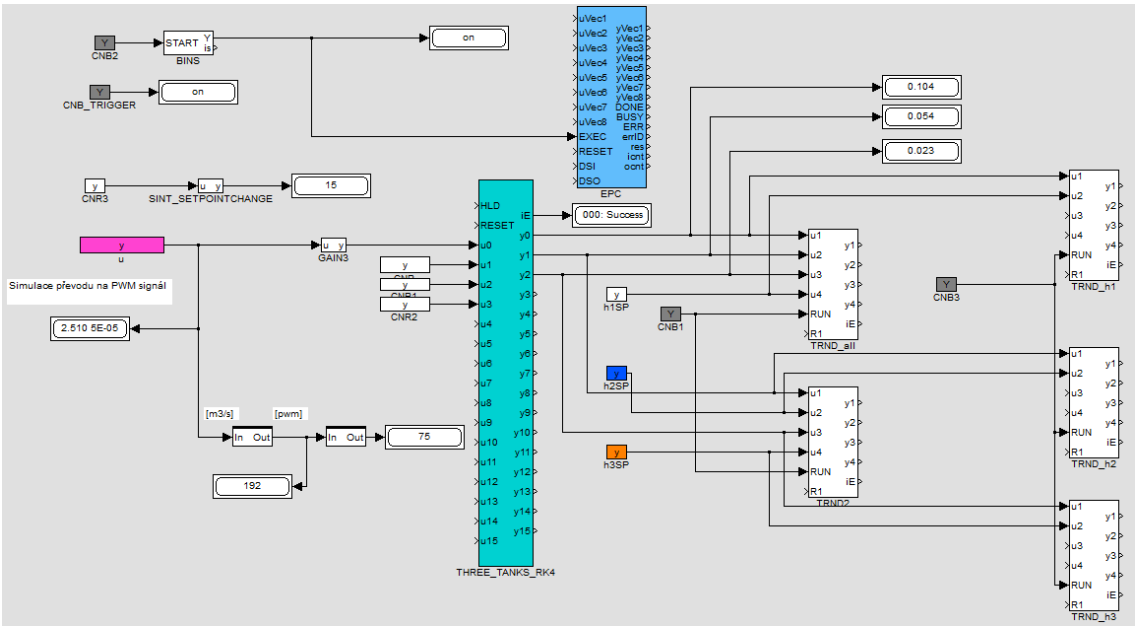
6.2 Simulace pomocí SIL

Tím že byl model úspěšně ověřen v simulaci MIL, lze přistoupit k další fázi simulace, kterou nazýváme SIL. V této fázi budeme pracovat v reálném čase s použitím řídicího systému Rexygen. V Matlabu zůstává zachován řídicí algoritmus „controller.m“, který zpracovává informace získané z matematického modelu, který nyní je součástí řídicího systému Rexygen. Po vyhodnocení všech dostupných informací „controller.m“ (dále jen controller) aplikuje vypočítaný optimalizovaný akční zásah zpět do řídicího systému Rexygen na matematický model a celý proces se v reálném čase opakuje v každém kroku jedné periody vzorkování. Komunikace mezi Matlabem a Rexygenem je vytvořena komunikačním rozhraní REST API využívající protokol HTTP. Na obrázku dole je zobrazeno principiální schéma, kde je vidět použití požadavků GET pro získání dat z cílového zařízení a použitím požadavku POST pro nastavení hodnot ve zdroji.



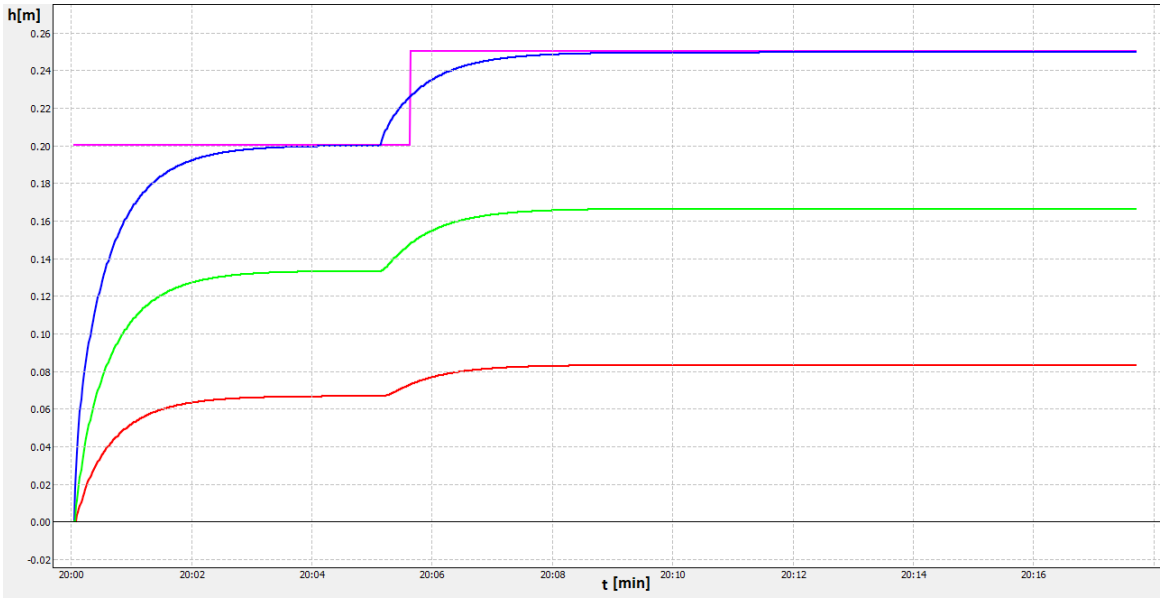
Obrázek 38: *Principiální schéma simulace SIL*

Výstup ze simulace získáme z funkčního bloku TREND, který slouží pro ukládání průběhů až čtyř vstupních signálů do cyklicky trendových bufferů v paměti cílového zařízení. Ukládání dat probíhá synchronně s během exekutivy reálného času, to umožňuje ukládat i velmi rychlé signály, tím nemůže dojít ke ztrátě dat ani jejich vícenásobného uložení. Řídicí algoritmus je zobrazen na obrázku. Matematický model je naprogramován ve volně programovatelném bloku REXLANG skriptovacím jazykem podobnému jazyku C. V algoritmu funkčního bloku pojmenovaném jako THREE_TANKS_RK4. REXLANG slouží pro vkládání funkcí, které nelze efektivně vytvořit z dostupné množiny bloků v knihovně programu.



Obrázek 39: *Řídicí algoritmus v Rexygenu*

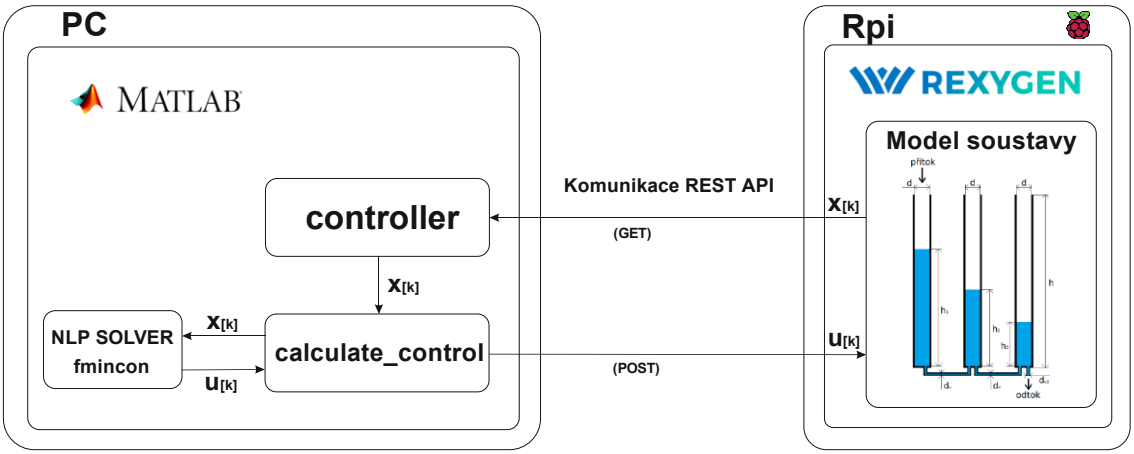
Z výstupní charakteristiky je vidět, že simulace se shoduje se simulací MIL. Získali jsme tím představu, zdali bude kód proveditelný i na cílovém zařízení. Pokud by se zaznamenaly výrazné rozdíly, museli bychom se vrátit k původní simulace a provést potřebné změny.



Obrázek 40: *Výstupní charakteristika ze simulace SIL*

6.3 Simulace pomocí PIL

Simulace PIL se od simulace SIL liší tím, že řídicí systém s modelem soustavy již není provozován na jednom počítači s controllerem, ale na cílové platformě Raspberry Pi, kde je spuštěno jádro RexCore. Tímto krokem lze zjistit, zdali je platforma schopna spustit řídicí algoritmus a jestli dokáže reagovat dostatečně rychle komunikace mezi controllerem a platformou Rexduino.



Obrázek 41: *Principiální schéma simulace PIL*

7 Implementace navržených řídicích algoritmů pro fyzikální model

7.1 Kalibrace senzorů pro reálný model

Při implementování navržených algoritmů, bylo zjištěno, že použité senzory neodpovídají údajům uvádějící v technickém dokumentaci výrobcem. Byla zjištěna výrazně horší přesnost, která je minimálně o jeden řád horší, než je uvedeno. Další problém byl zjištěn, při odměřování jednotlivých centimetrů výšky hladiny vůči měřenému napětí na vstupech platformy Rexduino. Změna napětí při odměřování sloupce kapaliny $0 - 31\,cm$ je velmi nelineární. Tyto důvody komplikují funkčnost regulátoru na reálné soustavě, proto sem se snažil je co nejvíce eliminovat.

Přesnost měření se bohužel eliminovat nepodaří, ale nelinearitu jsem se pokusil vyřešit pomocí funkčního bloku pro vyhodnocení polynomu (Funkční blok *POL*). Aby bylo možné tento blok použít, musíme nejdříve získat jednotlivé polynomy. K tomu byl použit Matlab, který disponuje nástrojem jménem *Basic Fitting*. Tento nástroj vygeneruje z naměřených dat vykreslených v Matlabu přes funkci *plot* polynomiální funkci, jejíž tvar a délku lze nastavit přímo v samotném nástroji. Získaný polynomy se následně nastaví do funkčního bloku *POL* v Rexygenu.

Popis funkčního bloku z knihovny Rexygen:

Funkční blok *POL* počítá hodnotu polynomiální funkce ve tvaru:

$$y = a_0 + a_1 u + a_2 u^2 + a_3 u^3 + a_4 u^4 + a_5 u^5 + a_6 u^6 + a_7 u^7 + a_8 u^8 \qquad (86)$$

Pro zajištění numerické robustnosti je polynom interně vyhodnocen pomocí Hornerova schématu.

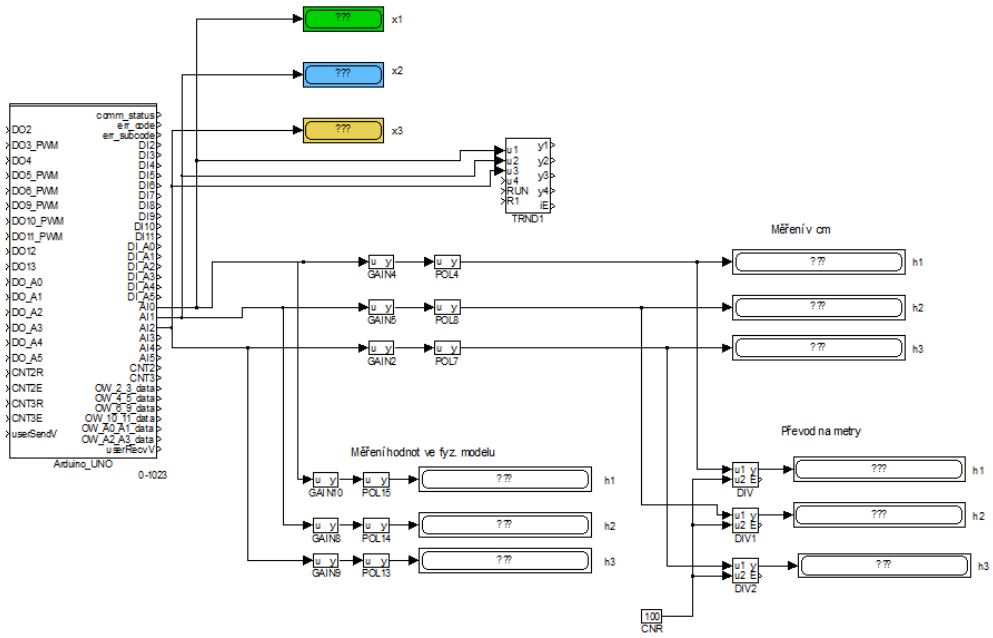
u – analogový vstupní signál

y – analogový výstupní signál

a_i – koeficient i-té mocniny vstupu, i = 0,1,…,8

Kalibrace senzorů

Kalibrace byla provedena pro všechny senzory stejným postupem. Byly změřeny hodnoty hladiny ve skleněné nádobě od $0 - 31\,cm$. V upraveném řídicím algoritmu (obrázek níže) se odečetly pro jednotlivé výšky naměřené hodnoty napětí vycházející z napěťového děliče. Všechna měření byla zapsána do tabulky, která jsou součástí elektroniké přílohy. Následně se z naměřených dat napsal kód pro zobrazení grafů v Matlabu a přes funkci *Basic Fitting*, se získala polynomiální funkci 4. řádu. Polynomiální funkce se zapsala do funkčního bloku POL, který při měření signálů ze vstupů Arduina nyní přibližně vypisuje alespoň orientační hodnoty pro řídicí algoritmus regulátoru. Dosažené a zpracované výsledky jsou zapsány v dalších podkapitolách:



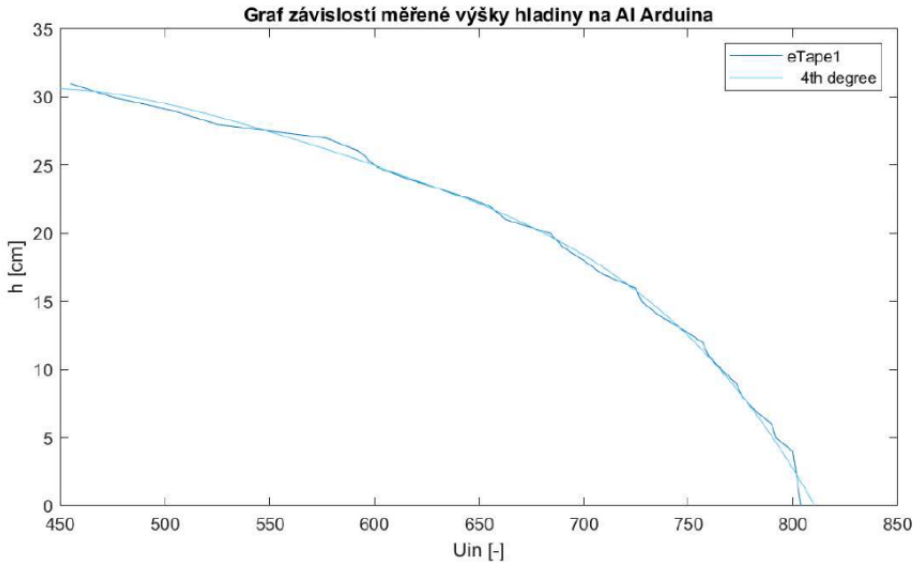
Obrázek 42: *Algoritmus pro odečítání naměřených hodnot senzorů*

Kalibrace senzoru pro měření první nádoby h_1

Získaná polynomiální funkce z nástroje *Basic Fitting* pro senzor h_1:

$$y = -392{,}41 + 2{,}9952.u - 0{,}0078251.u^2 + 8{,}9982.10^{-6}.u^3 - 3{,}9061.10^{-9}.u^4 \quad (87)$$

Vykreslený graf výstupní charakteristiky senzoru v závislosti měřené výšky na výstupním napětí z napěťového děliče je na obrázku níže.



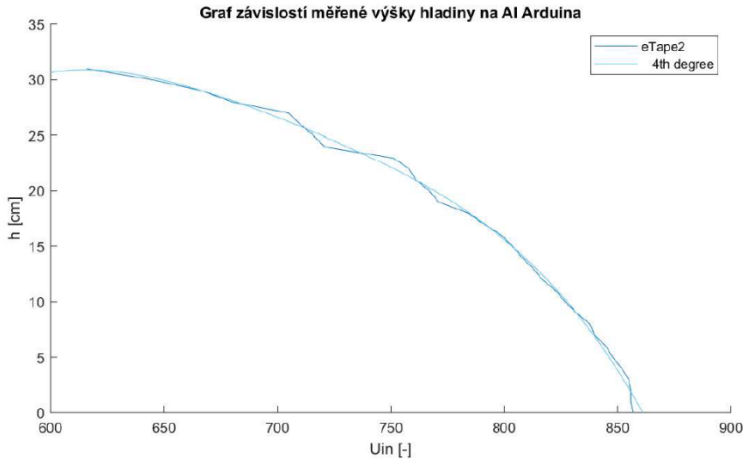
Obrázek 43: *Měření charakteristiky senzoru pro první nádrž h1*

Kalibrace senzoru pro měření první nádoby h_2

Získaná polynomiální funkce z nástroje *Basic Fitting* pro senzor h_2:

$$y = -3882{,}6 + 22{,}111.u - 0{,}046747.u^2 + 4{,}3941.10^{-5}.u^3 - 1{,}5552.10^{-8}.u^4 \quad (88)$$

Vykreslený graf výstupní charakteristiky senzoru v závislosti měřené výšky na výstupním napětí z napěťového děliče je na obrázku níže.



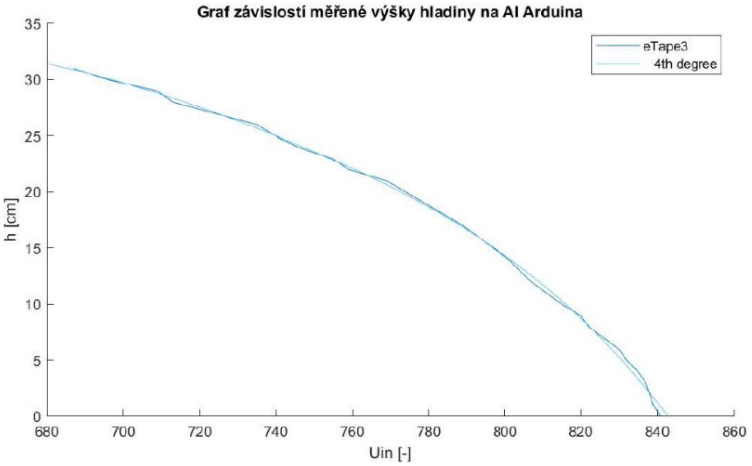
Obrázek 44: *Měření charakteristiky senzoru pro druhou nádrž h2*

Kalibrace senzoru pro měření první nádoby h_3:

Získaná polynomiální funkce z nástroje *Basic Fitting* pro senzor h_3:

$$y = -14878 + 81{,}041.u - 0{,}16506.u^2 + 0{,}00014949.u^3 - 5{,}0878.10^{-8}.u^4 \quad (89)$$

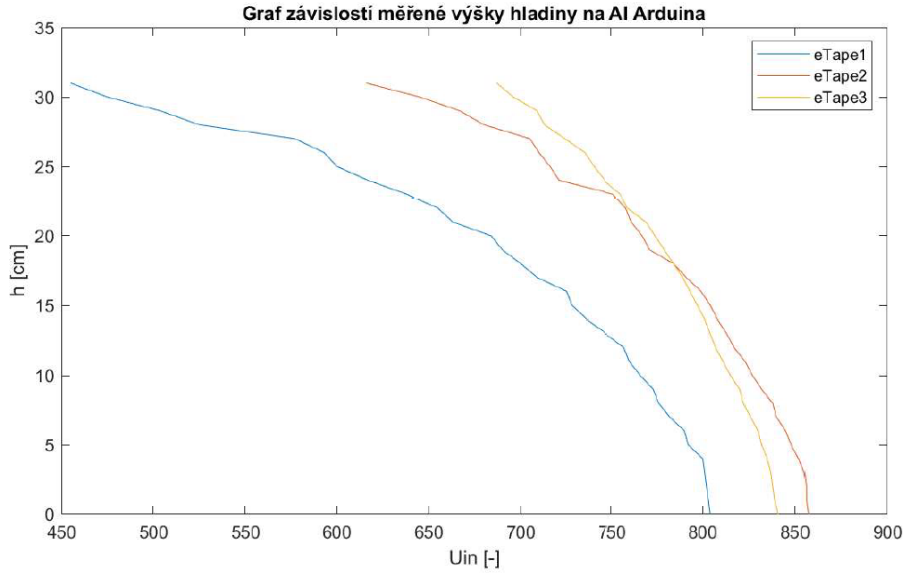
Vykreslený graf výstupní charakteristiky senzoru v závislosti měřené výšky na výstupním napětí z napěťového děliče je na obrázku dole.



Obrázek 45: *Měření charakteristiky senzoru pro třetí nádrž h3*

Porovnání výstupních charakteristik všech senzorů

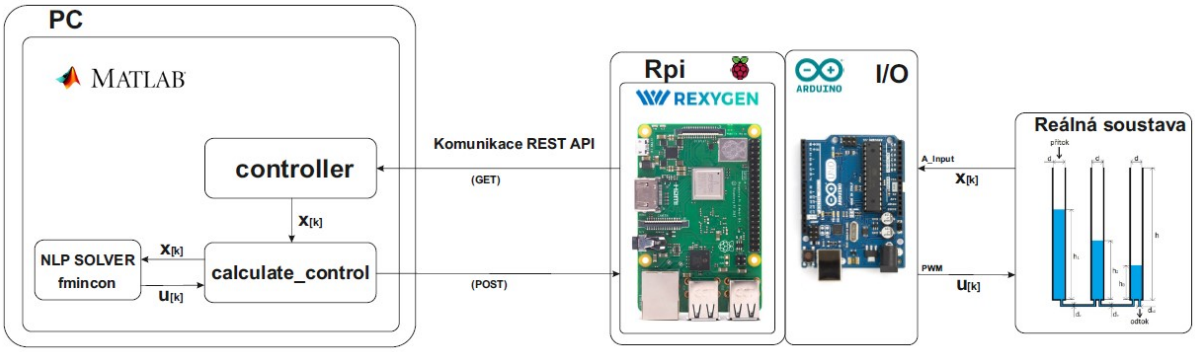
Na grafu změřených výstupních charakteristik, je vidět, že senzory se svým chováním od sebe dost liší. Proto je třeba při použití těchto senzorů provést měření a v rámci aplikování na proces separátně provést nastavení takové, aby alespoň orientačně odečítaly hodnoty pro řízený algoritmus. Při měření bylo zjištěno, že zejména v počátečních hodnotách $0 - 5\ cm$, rozlišení senzoru nedostatečně rozptyluje naměřené hodnoty, proto měření v této oblasti je velmi zkreslené.



Obrázek 46: *Porovnání měření mezi všemi senzory*

7.2 Řízení reálné soustavy

Algoritmus je realizován do cílové platformy Rexduino, kde Arduino funguje jako modul vstupů a výstupů, jak je vidět z principiálního schématu. Komunikace je stejně jako u simulace realizována přes REST API.



Obrázek 47: *Principiální schéma zapojení*

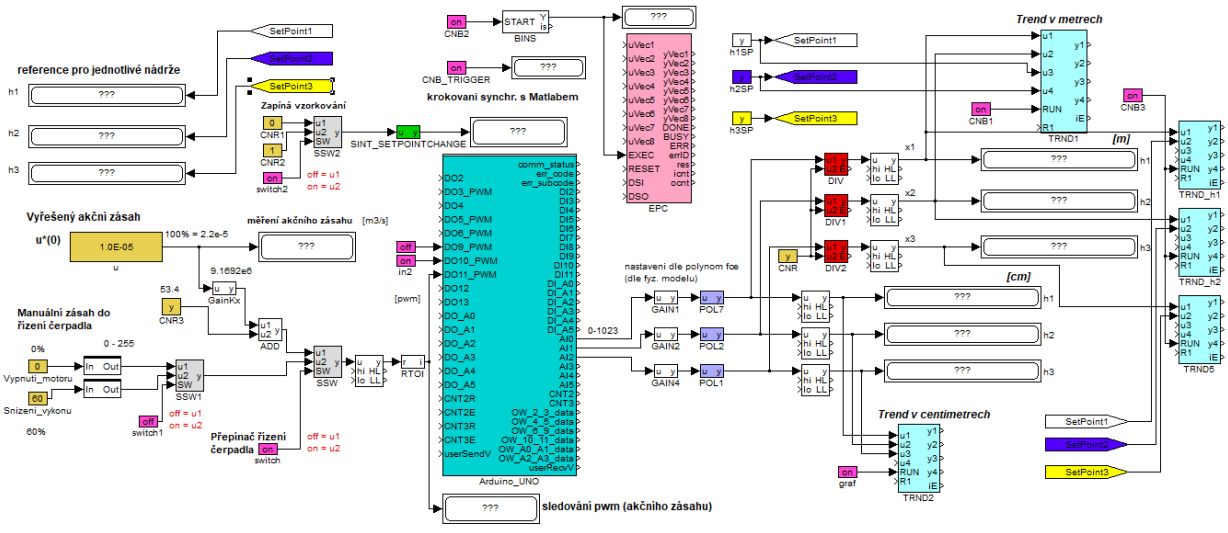
Na Raspberry Pi je nahrán řídicí algoritmus, který komunikuje s PC, na kterém je vypočítán ze získaných hodnot přes Matlab predikce optimálního akčního zásahu. Ta je poté opět zaslána do Raspberry Pi, kde se aplikuje hodnota na řízenou reálnou soustavu.

Řídicí algoritmus běžící na Raspberry Pi, je složen z funkčních bloků. Pro komunikaci s Arduinem, jako načítání a zapisování hodnot na vstupy či výstupy, slouží funkční blok *Arduino_UNO*. Tento blok není součástí samotného softwaru, ale je k dispozici na stránkách vývojářů ke stažení. Propojením ostatních funkčních bloků se vstupy nebo výstupy tohoto funkčního bloku se vytváří algoritmus. Funkční bloky jsou záměrně označeny barevně pro lepší přehlednost.

Další funkční blok důležitý pro realizaci je funkční blok s názvem *SINT_SETPOINTCHANGE*. Tento blok je propojen se skriptem v Matlabu a slouží jako realizace periody vzorkování. V knihovně jej najdeme pod názvem *SINT - jednoduchý integrátor*, realizuje diskrétní integrátor popsaný diferenční rovnicí. Uvnitř bloku je možné nastavit minimální a maximální meze, které jsou pro tuto práci nastaveny na $0 - 1000$ a integrační konstantu ti nastavenou na 1.

Pro odečítání hodnot ze senzoru jsem použil funkční bloky *POL* a pro převod na požadované měřené veličiny funkční blok *DIV*. Funkční blok *POL* jsem již zmiňoval při kalibraci, slouží pro vyhodnocení zadaného polynomu, pro zmírnění nelinearity senzorů. Funkční blok *DIV* je použit pro dělení dvou signálů, tak aby jednotky měřené hladiny byly v metrech. Měřené hodnoty jsou zaznamenávány funkčním blokem *TRND*, který slouží právě k ukládání až čtyřech vstupních signálů.

Většina funkčních bloků slouží ke zpracování nebo převod signálu, některé signály jsou pomocné pro rychlejší manipulaci při experimentování s fyzikálním modelem nebo pro vytvoření vizualizace.

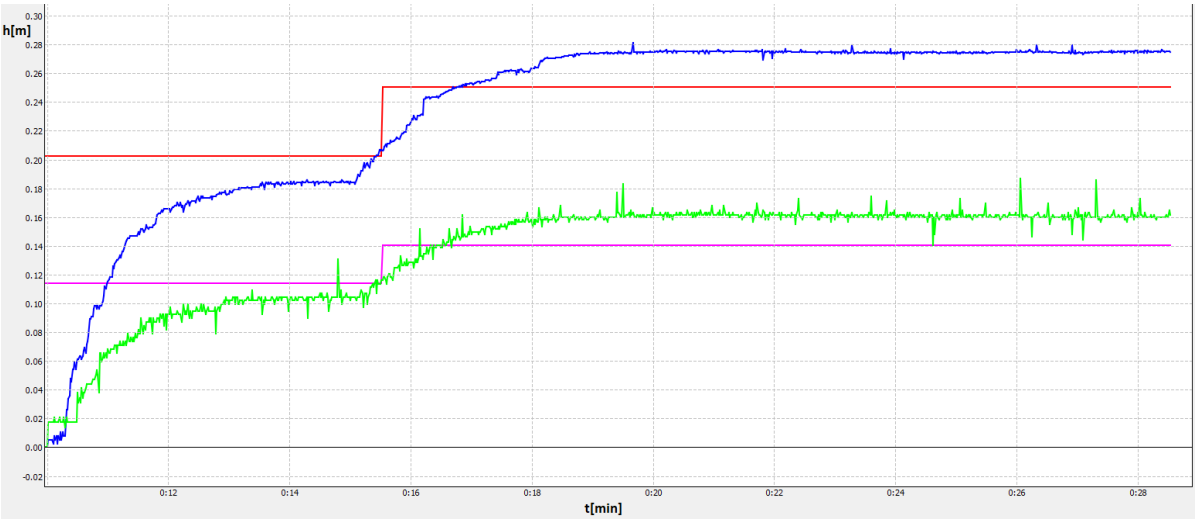


Obrázek 48: *Řídicí algoritmus v Rexygenu*

Na obrázku níže jsou znázorněna změřená data ze soustavy. Proběhlé měření na třetí nádrži h_3 nezobrazuji, z důvodu nepřesného měření. Je to způsobeno, tím že při velmi nízkých hladinách hladiny h_3 se v praxi setkáváme s dalším problémem, se kterým nepočítá zjednodušený matematický model

použitý v diplomové práci. Přesněji řečeno zjednodušený model nepočítá s fyzikálními veličinami týkající se přítoku tzn. s rychlostí a vztlakem. Přítok v h_3 tedy vyvolá při nízkých hladinách efekt, který se projevuje jako významné čeření vody, čímž znemožňuje dostatečně přesné měření ve třetí nádrži. Z těchto důvodů není zobrazen v průběhu.

Hladina h_1 a h_2 nejsou ideální jako při simulaci, avšak je vidět na grafu, že regulátor se snaží dosáhnout nastavené reference i když s určitým ofsetem. Tato nenulová regulační odchylka je způsobená špatnou kvalitou měření.

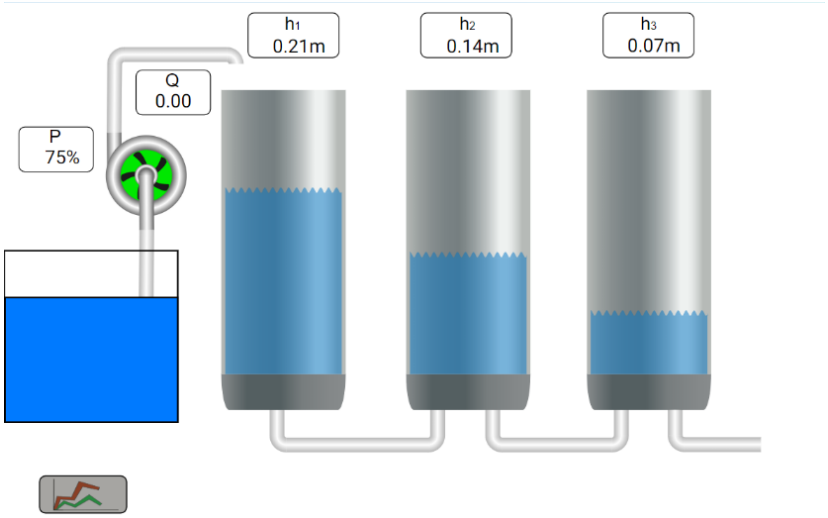


Obrázek 49: *Výstupní data z reálné soustavy*

7.3 Vizualizace

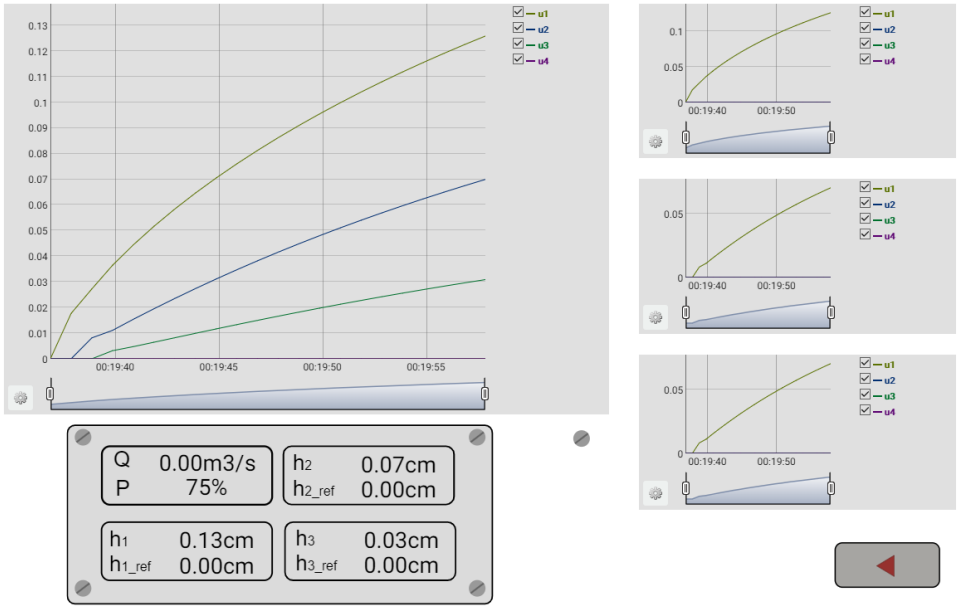
Vizualizace procesu je vtvořena pomocí HMI Designeru, který je součástí softwarového balíčku řídicího systému Rexygen. Bavíme se o HMI (Human Machine Interface), které představuje rozhraní mezi zařízením a člověkem. Jedná se o webovou aplikaci, která se vygeneruje po zkompilování řízeného algoritmu, kterého je součástí a komunikuje v reálném čase s běžícím algoritmem v jádru RexCore. Webová aplikace načítá data z jádra a na základě použitých komponent zobrazuje aktuální dění v řídicím algoritmu. Vizualizace se spustí nastavením IP adresy do webového prohlížeče, který je podporován pro spuštění jako například Google Chrome.

Vizualizace je sestavena ze dvou obrazovek. Jedna představuje zobrazení fyzikální soustavy a druhá zobrazuje trendy procesu. Použil jsem komponenty, které jsou součástí knihovny, navržené vývojáři. Tyto komponenty jsou propojeny pomocí cesty, která jednoznačně označuje cíl připojeného bodu k funkčnímu bloku a jejich proměnným. Data takto lze z řízeného systému buďto číst nebo je přímo ovládat vzdáleným přístupem. Vizualizace je interaktivní, takže na změnu hladiny nádrže reaguje i hladina ve vizualizace, to samé platí pro aktivní čerpadlo. Tlačítkem s průběhy přepneme na druhou obrazovku.



Obrázek 50: *Vizualizace první obrazovky*

Druhá obrazovka se skládá z komponent Trendů k zobrazení průběhů z průběhů řídicího algoritmu. Komponenta čte data z funkčního bloku *TRND* a zobrazuje stejné signály, které jsou zobrazovány v řídicím algoritmu.



Obrázek 51: *Vizualizace druhé obrazovky*

7.4 Návrhy na zvýšení kvality regulačního pochodu

Z důvodu prokázání špatných vlastností a nesprávné funkčnosti použitých senzorů při implementaci řídicího algoritmu na fyzikální model. Jsem hledal možnou alternativu, která by byla vzhledem k zjištěným problémům při implementaci, objektem zlepšení realizace řídicího algoritmu pro řízení modelu tří nádrží. To je ovšem jen jedno ze dvou opatření, čím zlepšit kvalitu regulačního pochodu. Druhým opatřením je zavedením zpětné vazby pro řízení čerpadla, v tuto chvíli jej ovládáme

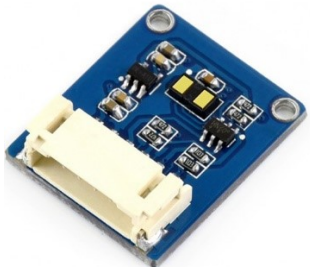
v otevřené smyčce, jinak řečeno posíláme na vstup PWM signál a předpokládáme, že tak průtok je takový, jaký chceme.

7.4.1 Doporučení pro úpravu senzoriky

Senzory jsou hlavním problémem. V této oblasti pro řízení soustavy je velmi důležité, aby zvolený způsob měření hladiny kapaliny byl přesný. To byl jeden z důvodů volby použitých senzorů eTape, u kterých byla uváděna přesnost $0{,}25\ mm$. Tento údaj však nebyl potvrzen. Problému se senzory bylo více, ale tomuto tématu se věnuji v kapitole 8. Jako alternativy pro senzoriku navrhuji následující senzory:

Laserový senzor vzdálenosti VL53L1X

VL53L1X je laserový zaměřovací modul nové generace ToF (Time-of-flight) umístěný v nejmenším možném pouzdře dostupném na současném trhu. Senzor vypočítává vzdálenost z prodlevy mezi vysláním a navrácením světelného paprsku. Je odolný vůči rušení a přesnost není ovlivněna odrazivostí povrchu. Umožňuje měřit absolutní vzdálenost do čtyř metrů bez ohledu na barvu cílového materiálu. ToF senzory se používají v asistenčních a bezpečnostních aplikacích pro pokročilé automobilové aplikace jako například detekce kolize. Další oblast použití je například robotika při mapování svého okolí či ve spotřebním zařízení jako jsou televizory nebo herní konzole.



Obrázek 52: *Senzor VL53L1X [27]*

Specifikace senzoru:

- Provozní napětí: 3,3 V / 5 V
- Rozměr: 20 mm × 24 mm
- Velikost montážních otvorů: 2,0 mm
- Rozsah vzdálenosti: 40 ~ 4000 mm
- Přesnost měření: ± 5%
- Zorné pole: 27 °
- Vlnová délka laseru: 940 nm
- Provozní teplota: -20 ~ 80 ° C

Připojení je realizováno přes 4 piny dva piny slouží pro napájení senzoru a další dva pro komunikaci přes sběrnici I2C. Jeden pin pro SDA a druhý pin pro SCL. SCL (seriál clock) je linka pro hodinový signál, zatímco linka SDA (seriál data) slouží pro obousměrný přenos dat.[27]

Sběrnice I2C

Je to sériová sběrnice tvořená dvouvodičovým datovým propojení mezi jedním nebo několika procesory a speciálními periferiemi součástkami. Princip sběrnice je založen na obousměrném plně duplexním přenosu dat, který je řízený společným hodinovým signálem vysílaným uzlem typu master a přijímán uzly typu slave. Všechna zařízení jsou připojena k jedné sběrnici a jsou cíleně vybírány svými adresami. Pokud jeden uzel vysílá, přijímají všechny ostatní a pouze podle adresy se určuje, který uzel data přijme. Přes I2C je možné připojit například senzory, LCD displeje nebo další digitálně řízené obvody.

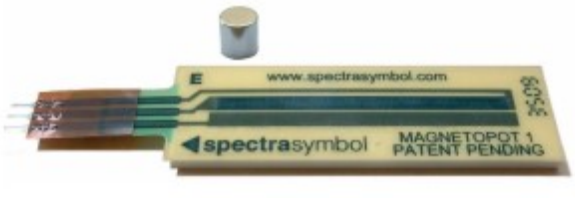
Aplikování na platformu Rexduino:

Řídicí systém Rexygen čtení dat přes I2C přímo nepodporuje. Pokud bychom chtěli použít tento typ senzoru, museli bychom zasáhnout do zdrojového kódu nahraného v Arduinu nebo přes programovatelný funkční blok REXLANG naprogramovat alternativu za funkční blok pro komunikaci s Arduinem. To by ovšem znamenalo výrazný zásah do řešení řídicího systému, proto zvolením této varianty volíme i velmi náročné řešení na zprovoznění.

Obrázek 53: *Principiální schéma propojení komunikace*

Senzor MagnetoPot

MagnetoPot je bezkontaktní lineární potenciometr použitelný pro kontinuální měření hladiny kapaliny. Tento inovativní senzor umožňuje konstrukčně zapojit na vnější stranu nádoby a bezkontaktním způsobem měřit magnet uvnitř nádoby. To by znamenalo navrhnout plovák s magnetem, který by při stoupání či klesání hladiny ovlivňoval senzor a určoval výšku hladiny. Po celou dobu se tak senzor vyhýbá kontaktu s kapalinou a poskytuje lineární výstup. Funguje jako dělič napětí.[28]

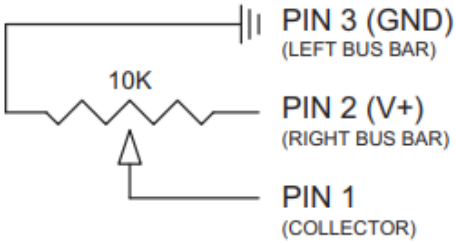


Obrázek 54: *Senzor MagnetoPot [28]*

Aplikování na platformu Rexduino:

Senzor by byl jednoduše připojen ke vstupům a výstupům Arduina přes 3 vodiče. Dvěma vodiči by se realizovalo napájení senzoru a třetím vodičem by se odečítaly hodnoty. Senzor se na první pohled

jeví svou funkčností jako senzory aplikované v diplomové práci, avšak rozdíl a tím i jeho velkou výhodou je to, že magnet v plováčku spíná pouze v místě, kde se nachází a kde tedy působí svou magnetickou silou. Zatímco u senzorů eTape, které využívají pro měření změnu hydrostatického tlaku, který na ně působí, dochází k výrazným nepřesnostem i při nepatrné změně působící na obal senzoru.



Obrázek 55: *Schéma zapojení [28]*

Ultrazvukový senzor HC-SR04

Ultrazvukový měřič vzdálenosti pro Arduino. Senzor, který disponuje jedním vysílačem a jedním přijímačem. Modul umožňuje spolehlivou detekci v rozmezí $2-400\,m$, nejpřesnější je však v prvních dvou metrech. Často se používá u robotů a na místech, kde je potřeba s velkou přesností zjišťovat překážky před senzorem. Přesnost senzoru je udávána $3\,mm$, k propojení slouží 4 vodiče, dva k napájení a dva k čtení dat ze senzoru. Vodič *Trig* spínáme vysílač senzoru po dobu $5\,\mu s$, ultrazvukový vysílač vyšle vysokofrekvenční pulz a poté čekáme na zachycení odrazu ultrazvukovým přijímačem přes vodič *Echo*. Data zpracujeme tak, že přečtenou délku pulzu pomocí výpočtu převedeme na vzdálenost v centimetrech.

Obrázek 56: *Senzor HC-SR04 [33]*

Aplikování na platformu Rexduino:

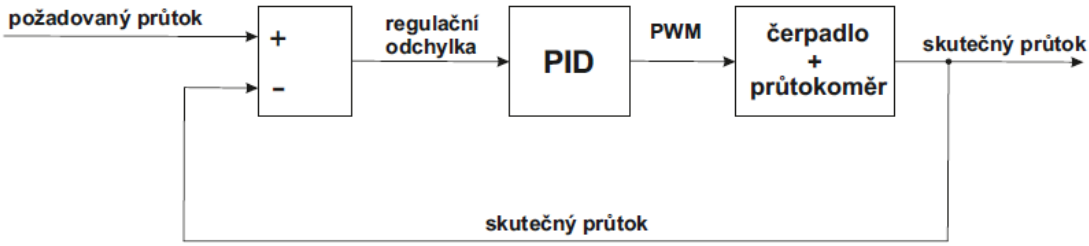
Podobně jako u laserového senzoru VL53L1X, bude aplikování náročnější z důvodů, že pro tento senzor není přímá podpora od vývojářů, tudíž by realizace bylo možná v rámci zásahu do zdrojového kódu nahraného v Arduinu. Další možností je počkat na vydání novější verze Rexygenu, jelikož po konzultaci s vývojáři byla přislíbena do budoucna podpora pro tento typ senzoru. Za zmínku jistě stojí i možnost použití dalšího Arduina, provádějící zpracování hodnot ze senzoru, které by posílalo do PC a dále přes komunikační rozhraní REST API by se zasílala data do Raspberry Pi. Zvolení této varianty není příliš praktické, proto jej nebereme v úvahu.

Doporučení pro aktuátor

Další doporučení, jak zvýšit kvalitu regulačního pochodu je zavedení pomocné regulační smyčky na řízení čerpadla a to tak, že průtok čerpadla budeme měřit průtokoměrem pro nízký průtok.

Obrázek 57: *Navrhovaný průtokoměr [29]*

Průtokoměr se zavede za výstup čerpadla a PID regulátor se bude starat o to, aby skutečný průtok čerpadla byl stejný jako požadovaný, tak aby regulační odchylka byla nulová.



Obrázek 58: *Pomocná regulační smyčka*

8 Závěr

Cílem diplomové práce bylo seznámení se s problematikou NMPC, která není běžnou součástí výuky a seznámení se k tomu potřebnými nástroji a metodami. Pro prokázání získaných znalostí byl navržen a realizován vhodný algoritmus nelineárního modelu prediktivního řízení pro předem definovanou soustavu tří nádrží. Konstrukce tohoto fyzikálního modelu soustavy, na kterém bylo třeba realizovaný algoritmus demonstrovat, vyžadovalo zvolení vhodných aktuátorů, senzoriky a jiných elektronických komponent, včetně řídicí platformy. Řídící platforma byla zvolena kombinace propojení Raspberry Pi s Arduino UNO s řídicím systémem Rexygen, jinak nazývanou Rexduino. I díky volbě tohoto řídicího systému, jsme před samotnou implementací na reálny model provedly otestování funkčnosti na simulaci MIL, SIL, PIL.

Práce je rozložena do kapitol, které následují po sobě dle dílčích úkolů z oficiálního zadání, tak jak jdou po sobě a jak jsem postupoval po celou dobu zpracování tématu. V podstatě první dvě kapitoly se týkají teoretické části a seznámení s problematikou nelineárního prediktivního řízení a použitých metod při řešení problému optimalizačního řízení.

Kapitola, která následuje po teoretické části se zbývá popisem fyzikálního modelu se strukturálním rozložením použitých elektronických komponent. Tyto elektronické komponenty jsou popsány a popisuji realizování pro senzoriku a aktuátory. Konkrétně pro senzoriku podle jednoduchosti a dostupné funkčnosti pro zvolený fyzikální soustavu, jsem zvolil odečítání přes napěťový dělič. Pro řízení čerpadla jsem zvolil již realizovaný H-můstek pro řízení stejnosměrných motorů, který lze běžně zakoupit ve vybraných obchodech s modelářskými komponenty. Dále zde popisuji funkční spojení řídicí platformy Rexduino s popisem řídicího systému Rexygen, s kterým bylo třeba se plně seznámit před aplikováním na vybranou úlohu.

V kapitole návrhu a realizaci prediktivního řízení je podstata diplomové práce, kde se zaměřuji na konkrétní návrh prediktivního řízení, který je odvozen od samotného popisu matematického modelu, vycházejícího ze známe problematiky řízení systému tří nádrží, který je transformován do diskrétního tvaru metodou RungeKutta 4. řádu. Regulátor je realizován pomocí skriptu vytvořeném v Matlabu a pro řešení optimalizační úlohy je použitá funkce fmincon, která je součástí optimalizačního toolboxu Matlabu. Do této kapitoly je zahrnuta i identifikace soustavy, kterou jsem použil pro identifikování parametrů, důležitých pro realizování dostatečně přesného modelu použitého v realizovaném prediktivním regulátoru.

Před implementováním regulátor na fyzikální model, jsem provedl simulace typu MIL, SIL a PIL, těmito simulacemi se potvrdila správnost řešení navrženého regulátoru. Výstupy z jednotlivých simulací byly mezi sebou totožné a na základě těchto simulací bylo možné i s regulátorem experimentovat a ladit parametry regulátoru pro zlepšení přesnosti řízení.

Při implementování algoritmu na fyzikální model, došlo k nesouladu výstupů měřených údajů s údaji z výstupu ze simulací. Tudíž kvalita regulačního pochodu při regulaci je horší, než jsem předpokládal. To je způsobeno hned několika faktory. Jeden z nich je, že při návrhu senzorů, doporučených vedoucím práce a na základě technické dokumentaci uváděné výrobcem, se nepočítalo se špatnými vlastnostmi senzoru. Přesnost senzoru je o minimálně jeden řád horší, než výrobce udává

a samotný senzor vykazuje nelineární výstup, ten jsem se snažil potlačit při návrhu řídicího algoritmu v Rexygenu. S přesností senzoru jsem se bohužel nevypořádal, proto výstup z regulace fyzikálního modelu ukazují velké nepřesnosti. Dalším faktorem bylo zjištění vysoké citlivosti dynamických vlastností soustavy zejména na průřezu odtokové trubice ze soustavy, ty jsem se pokusil eliminovat tím, že se výstup zafixoval do ustálené pozice. Eventuální změna hardwarových komponent, zejména v části senzoriky, by představovala významný zásah překračující rámec diplomové práce. Proto po dohodě s vedoucím práce jsem do práce zahrnul podkapitolu uvádějící možné návrhy na změnu senzoriky či vylepšení aktuátoru.

Z časových důvodů se již nepřešlo k změně senzorů ani k realizování jiných opatření pro zvýšení kvality regulačního pochodu. Proto jsem alespoň sepsal doporučení, či alternativy, které by měly pomoci při budoucí realizaci nebo pro použití tématu v jiné podobné akademické práci.

V samotném závěru, i když to nebylo v samotném zadáním diplomové práce, jsem vytvořil pomocí nástroje HMI designer vizualizaci pro lepší prezentování například ve výuce. Vytvořil jsem ji pomocí komponent, které jsou součástí nástroje HMI designer a který je zase součástí softwarového balíčku řídicího systému Rexygen.

Použitá literatura

[1] QIN, S.Joe a Thomas A. BADGWELL. A survey of industrial model predictive control technology. Control Engineering Practice [online]. 2003, 11(7), 733-764 [cit. 2020-04-04]. DOI: 10.1016/S0967-0661(02)00186-7. ISSN 09670661. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0967066102001867

[2] NOVÁK, Martin. Nelineární Prediktivní Řízení. Zlín, 2014. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Vedoucí práce Petr Chalupa.

[3] RUCHIKA a Neha RAGHU. Model Predictive Control: History and Development. International Journal of Engineering Trends and Technology [online]. Thapar University, Patiala: Department of Electrical and Instrumentation Engineering, 2013, 6 June 2013, , 2600-2601 [cit. 2020-04-04]. ISSN 2231-5381. Dostupné z: http://ijettjournal.org/volume-4/issue-6/IJETT-V4I6P173.pdf

[4] BAO-CANG, Ding. MODERN PREDICTIVE CONTROL. Taylor & Francis Group 6000 Broken Sound Parkway NW, Suite 300: CRC Press, 2010. ISBN 978-1-4200-8530-3.

[5] CAMACHO, E.F. a C. BORDONS. Model Predictive control. Second Edition. London: Springer, 2007. ISBN 978-1852336943.

[6] SUNAN, Huang, Tan KOK KIONG a Lee TONG HENG. Applied Predictive Control. London: Springer, 2002. ISBN 978-1-84996-864-5.

[7] KOUVARITAKIS, Basil a Mark CANNON. Nonlinear Predictive Control: theory and practice. London: The Institution of Engineering and Technology, 2001. ISBN 978-0-85296-984-7.

[8] MAGNI, Lalo, Frank ALLGÖWER a Davide M. RAIMONDO. Nonlinear Model Predictive Control: Towards New Challenging Applications. Berlin: Springer, 2009. ISBN 978-3-642-01093-4.

[9] GRANCHAROVA, Alexandra a Tor Arne JOHANSEN. Explicit Nonlinear Model Predictive Control: Theory and Applications. Berlin: Springer, 2012. ISBN 978-3-642-28779-4.

[10] VELECKÝ, Pavel. Predikce v prediktivním řízení. Zlín, 2010. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Marek Kubalčík.

[11] Performance of sinusoidal scanning with MPC in AFM imaging: Discrete model predictive control approach. In: Research gate [online]. 2014 [cit. 2020-04-21]. Dostupné z: https://www.researchgate.net/publication/262388793_Performance_of_sinusoidal_scanning_with_MPC_in_AFM_imaging

[12] HÝL, Radim. Návrh a implementace prediktivního řízení. Ostrava, 2017. Disertační práce. Vysoká škola báňská - Technická univerzita Ostrava, Fakulta strojní. Vedoucí práce Renata Wagnerová.

[13] MIŠENČÍK, Patrik. Řízení laboratorní soustavy prediktivním regulátorem s uvažováním omezení. Pardubice, 2014. Diplomová práce. Univerzita Pardubice, Fakulta elektrotechniky a informatiky. Vedoucí práce Daniel Honc.

[14] MIKULÁŠ, Ondřej. A Framework for Nonlinear Model Predictive Control. Praha, 2016. Diplomová práce. Czech Technical University in Prague. Vedoucí práce Ondřej Šantin.

[15] DAI, Li, Yuanqing XIA a Mengyin FU, MAHMOUD, Magdi S., ed. Advances in Discrete Time Systems: Discrete-Time Model Predictive Control [online]. China: School of Automation, Beijing Institute of Technology, 2012 [cit. 2020-04-26]. ISBN 978-953-51-5720-5. Dostupné z: https://www.intechopen.com/books/advances-in-discrete-time-systems/discrete-time-model-predictive-control

[16] PREJZEK, Michal. Návrh a realizace regulačního obvodu na fyzikálním modelu tří nádrží pomocí kompaktního programovatelného automatu WinPac-8000. Ostrava, 2010. Diplomová práce. Vysoká škola báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. Vedoucí práce Štěpán Ožana.

[17] ETape Continuous Fluid Level Sensor Operating: Instructions and Application Notes [online]. New Jersey: Milone Technologies [cit. 2020-04-27]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/eTapeApp.pdf

[18] MICHÁLEK, Ondřej. Realizace výukové úlohy z oblasti automatického řízení na platformě REXduino. Ostrava, 2017. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. Vedoucí práce Štěpán Ožana.

[19] Arduino Introduction. What is Arduino? [online]. [cit. 2020-05-03]. Dostupné z: https://www.arduino.cc/en/Guide/Introduction

[20] Raspberry Pi: Úvod a výběr desky. Arduino návody [online]. 2018, 5. 6. 2018 [cit. 2020-05-03]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/raspberry-pi-uvod-a-vyber-desky.html?gclid=Cj0KCQjw17n1BRDEARIsAFDHFezbmxWOPVuXaQx-uc_YvF4k2CJUmcixveoL2NLgzdmQDEpwb89iOc4aAhOpEALw_wcB

[21] Raspberry Pi: Raspberry Pi 3 Model B [online]. [cit. 2020-05-03]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[22] REX controls. REX controls: Dokonalé spojení Raspberry Pi a mikrokontroléru Arduino [online]. Plzeň, 2014, 26. 8. 2014 [cit. 2020-05-03]. Dostupné z: https://www.rexcontrols.cz/clanky/dokonale-spojeni-raspberry-pi-a-mikrokontroleru-ar

[23] VOJÁČEK, Antonín. Principy průmyslových čerpadel: 6.díl - pístová čerpadla. Automatizace.hw [online]. 2011, 26. Únor 2011 [cit. 2020-05-04]. Dostupné z: https://automatizace.hw.cz/principy-prumyslovych-cerpadel-6dil-pistova-cerpadla

[24] Funkční bloky systému REXYGEN [online]. plzeň: REX Controls, 2019 [cit. 2020-05-04]. Dostupné z: https://www.rexygen.com/doc/CZECH/MANUALS/BRef/BRef_CZ.html

[25] Rapid development, RT simulace, modelování [online]. Plzeň: REX Controls, c2000-2020 [cit. 2020-05-04]. Dostupné z: https://www.rexcontrols.cz/rapid-development-rt-simulace-modelovani

[26] BOBÁL, Vladimír. Adaptivní a prediktivní řízení. Zlín: Univerzita Tomáše Bati ve Zlíně, 2008. ISBN 978-80-7318-662-3.

[27] Waveshare: VL53L1X Distance Sensor [online]. 2018 [cit. 2020-05-06]. Dostupné z: https://www.waveshare.com/wiki/VL53L1X_Distance_Sensor

[28] In: Spectrasymbol: MAGNETOPOT [online]. [cit. 2020-05-07]. Dostupné z: https://media.digikey.com/pdf/Data%20Sheets/Spectra%20Symbol/MP1%20Series%20MagnetoPot.pdf

[29] In: Made in china [online]. [cit. 2020-05-07]. Dostupné z: https://m.made-in-china.com/product/100ml1L-Min-Low-Flow-Rate-Water-Dispenser-Flow-Measurement-Sensor-884120905.html

[30] Getting started with REXYGEN and Raspberry Pi User guide. Getting started with REXYGEN and Raspberry Pi User guide [online]. REX Controls, 2019, 2019-04-20 [cit. 2020-05-12]. Dostupné z: https://www.rexygen.com/doc/ENGLISH/MANUALS/RexygenGettingStarted_RasPi/RexygenGettingStarted_RasPi_ENG.html#x1-20001

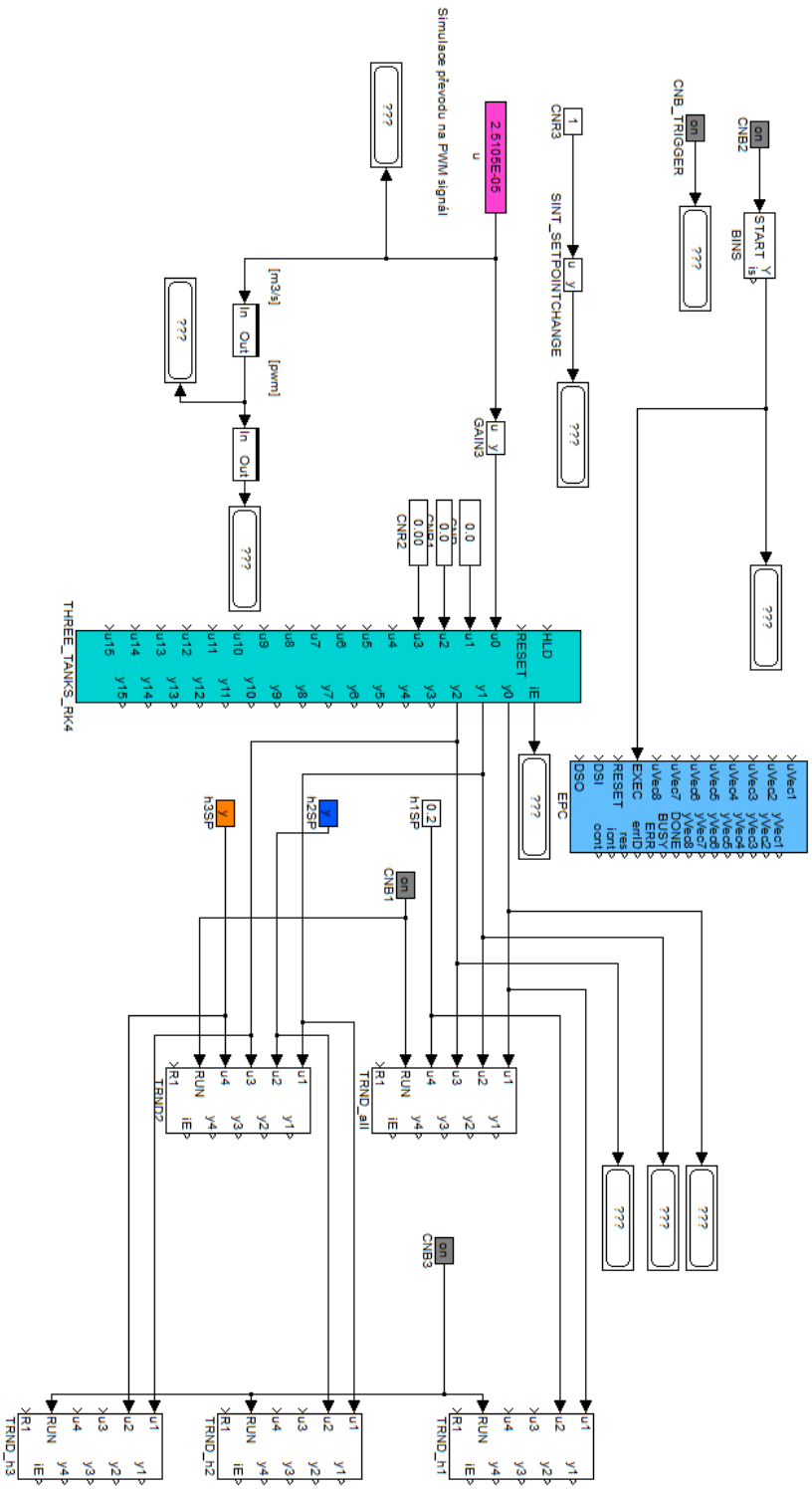
[31] How To Mechatronics: Arduino DC Motor Control Tutorial – L298N [online]. howtomechatronics [cit. 2020-05-12]. Dostupné z: https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

[32] VÍTEČEK, Antonín a Miluše VÍTEČKOVÁ. Optimální systémy řízení. Ostrava: VŠB-Technická univerzita, 1999. ISBN 978-80-248-4239-4.

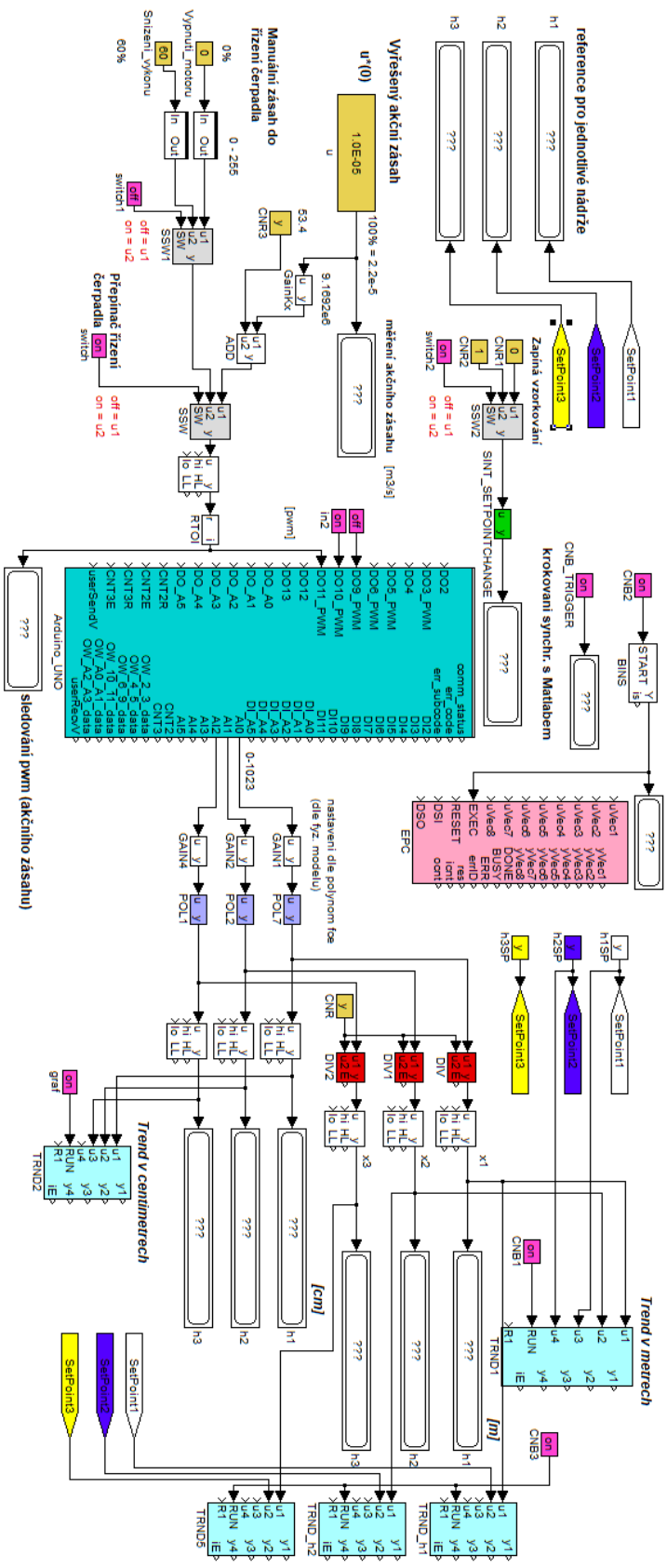
[33] Instructables circuits. Instructables circuits: Easy Ultrasonic 4-pin Sensor Monitoring [online]. [cit. 2020-05-14]. Dostupné z: https://www.instructables.com/id/Easy-ultrasonic-4-pin-sensor-monitoring-hc-sr04/

Seznam příloh

Příloha I: Řídicí algoritmus pro SIL v prostředí REXYGEN



Příloha II: Řídicí algoritmus pro PIL v prostředí REXYGEN



Příloha III: CD-ROM disk:

Součástí této diplomové práce je elektronická příloha, která obsahuje:

- Diplomovou práci ve formátu PDF
- Adresář Model soustavy – soubory s modelem soustavy
- Adresář Identifikace – soubor se skripty pro identifikaci včetně s měřenými daty
- Adresář Kalibrace senzorů – soubor se skriptem pro kalibraci senzorů s měřenými daty
- Adresář Simulace MIL – soubor se skriptem pro simulaci MIL
- Adresář Simulace SIL – soubor se skriptem controlleru a řídicím algoritmem v Rexygenu včetně vizualizace
- Adresář Algoritmus pro fyzikální model – soubor se skriptem controlleru a řídicím algoritmem v Rexygenu včetně vizualizace
- Záznam s ukázkou vizualizace v simulaci SIL
- Záznam s návodem pro provedení parameter estimation